MICROCOMPUTADOR CURSO PRÁTICO

TIRE O MÁXIMO DO MICRO EM 30 SEMANAS

2

Redes: aplicações e recursos
Manipulação de arquivos

rioGráfica

MICROCOMPUTADOR CURSO PRÁTICO

APLICAÇÕES

Informação em cadeia 37

PERIFÉRICOS

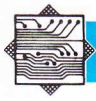
Dados na tela 40

SOFTWARE

Memória organizada 42

PROJETOS DE PROGRAMAS

Somas mágicas 44


RAIO X

APPLE II*c* 46

CIÊNCIA DA COMPUTAÇÃO

A estrutura dos blocos de adição 48

LINGUAGENS

A arte da tartaruga 50

TÉCNICAS DE PROGRAMAÇÃO

Cuidados com o estilo 54

PERFIL

Os dissidentes 56

TESTE

Depois de ler com atenção todo o fascículo, responda a estas perguntas. Veja a solução no próximo número, junto com um novo teste.

1) Quantos slots para encaixe de periféricos tem o Apple II*c*?
2) Por que não se pode construir um quadrado mágico de 4 x 4?
3) Por que não se deve usar num mesmo programa em BASIC duas variáveis chamadas CONT e CONTADOR?
4) Quais são os tipos de arquivo, em relação à maneira de armazená-los e acessá-los?
5) Qual é a unidade de medida da resolução gráfica de um monitor de vídeo?
6) Diga se está correto este procedimento LOGO para desenhar um triângulo eqüilátero:
   REPITA 3 [FR 50 DI 60]

RESPOSTAS DO TESTE ANTERIOR

1) Os dados num cartucho são gravados num circuito integrado, que pode ser do tipo PROM (não apagável) ou EPROM (apagável).
2) O IBM PC endereça os dados num sistema de 16 bits, e não 8, como a maioria dos micros.
3) Softwares integrados são aqueles que combinam recursos para várias aplicações: processamento de texto, planilha financeira, banco de dados, geração de gráficos e, em alguns casos, telecomunicações.
4) As três operações principais da lógica booleana são E, OU e NÃO.
5) A linguagem que deu origem ao LOGO foi o LISP.
6) O comando LOGO que faz a tartaruga andar sem traçar linhas é SEMCOR (abreviado SC).

PLANO DA OBRA

MICROCOMPUTADOR - CURSO PRÁTICO é uma coleção de 31 fascículos de periodicidade semanal. Os trinta primeiros terão, cada um, vinte páginas internas (miolo) e quatro capas, além de uma introdução de dezesseis páginas acompanhando o fascículo 1. O fascículo 31 terá doze páginas e quatro capas, e incluirá o índice geral da obra. Os miolos, encadernados, constituem dois volumes de um curso prático de microcomputação, com seções indicadas por tarjas de diferentes cores.
As duas primeiras capas são descartáveis; as 3.ª s e 4.ª s trazem programas de jogos. Encadernadas, formam um volume de cem páginas.

COMO ENCADERNAR

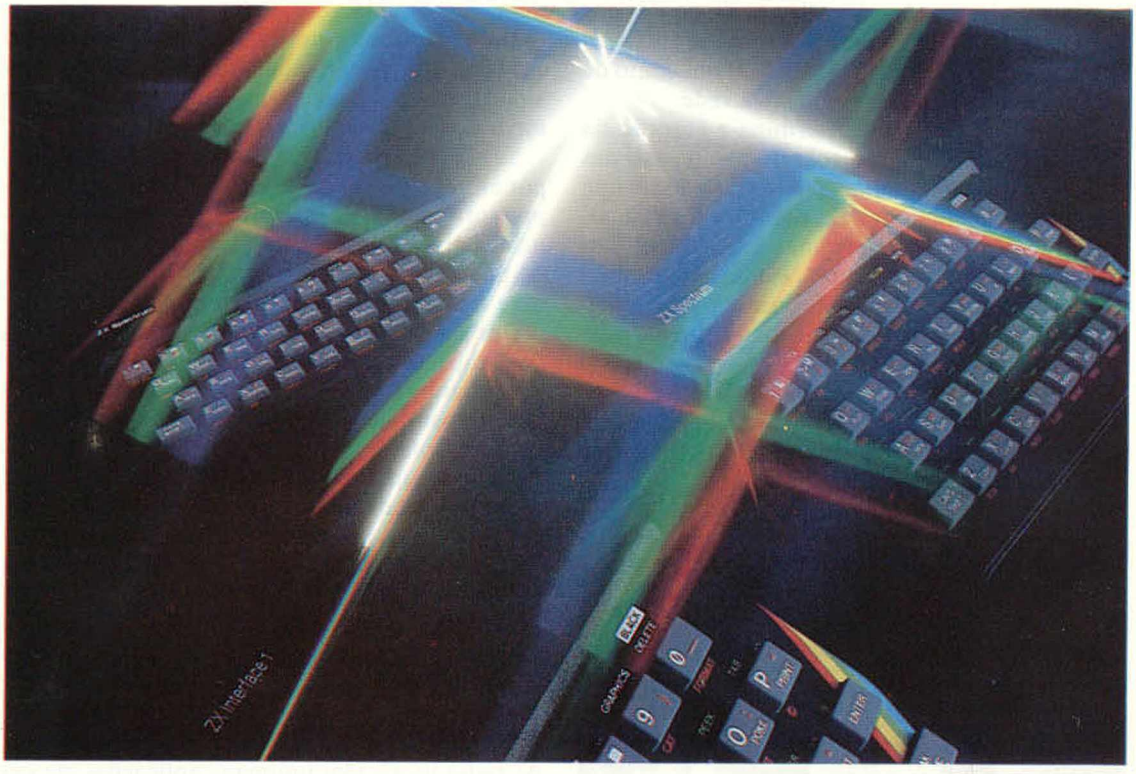
As capas duras, incluindo as guardas, estarão à venda simultaneamente com os fascículos 10 (**Volume 1**) e 25 (**Volume 2** e **Jogos**).
Volume 1 - Deve ser encadernado com os elementos dispostos nesta ordem: guardas; frontispício e introdução (publicados com o fascículo 1); miolos dos fascículos de 1 a 15; e guardas.
Volume 2 - Guardas; frontispício (que virá com o fascículo 31); miolos dos fascículos de 16 a 31; e guardas.
Jogos - Guardas; frontispício e sumário (que constituem as 3.ª s e 4.ª s capas do fascículo 31); 3.ª s e 4.ª s capas dos fascículos de 1 a 30; e guardas.

rioGráfica **Diretoria Executiva:** Oscar Neves; Filipe Zander; Nilo Sérgio de Almeida; Danilo Esteves Costa; **GRUPO EDUCAÇÃO E CULTURA - Planejamento e Comercialização - Diretor:** Roberto Combochi; **Gerente de Marketing:** Jaime Rodrigues; **Gerente de Produto:** José Mauro Porto; **Gerente do P.C.P.:** Aylton Menezes; **Gerente de Vendas:** Rubens Barbosa; **Coordenadores de Promoção:** Paulo Cesar M. Seixas; Edgar Mello M. Neto; **Gerente de Circulação:** Norberto Martin; **Supervisor de Operações:** Abel Pereira Pinto - **Editorial - Diretor:** João J. Noro; **Editor-chefe:** Mauricio Rittner; **Administração Editorial:** Heitor de Souza Paixão; **Editor:** Jesse Navarro; **Revisores:** Isis Augusta Loyolla; Cacilda Guerra; Luiz Vicente Vieira Filho; Marcos Domingos Agathão; Maria da Graça Mendonça Couto; **Diretor de Arte:** Aníbal Monteiro; **Chefe de Arte:** Bonifácio Duardes Miranda; **Chefe de Estúdio:** José Yuji Kuribayashi; **Assistente de Arte:** Miguel Luis Escámez Simón - **Colaboradores - Editores-assistentes:** Isa Mara Lando; Luiz Carlos Pizarro Marin; **Texto:** Jussara Maturo; **Preparação de Texto:** Luiz Carlos Cardoso; **Tradução:** Cláudio Marcondes; Maria Clara Cescato; Patrícia de Paiva e Castro; Regina Carmo R. Silva; **Pesquisa Iconográfica:** Stella Maria Quentel; **Arte:** Toshio Morobosi; **Assessoria Técnica:** Derval Junqueira de Aquino; Lauro de Lauro; Luiz Henrique Alayon. As fotos não creditadas pertencem à obra original. Segue-se o crédito das demais: Fernando Scavone. © APSIF Copenhagen, 1984; © Orbis Publishing Ltd., 1984; © Editora Rio Gráfica Ltda., 1985, para a língua portuguesa. **Composição:** ERG Ltda. e AM Produções Gráficas Ltda. **Impressão:**

NÚMEROS ATRASADOS - Você poderá comprar os exemplares que faltam em sua coleção, pelo preço do último fascículo posto à venda, em sua banca de jornal preferida. Caso não consiga obtê-los, faça sua solicitação por carta endereçada diretamente à Editora Rio Gráfica Ltda., rua Itapiru, 1209, Rio Comprido, Rio de Janeiro, CEP 20251. O atendimento será feito por via postal. Nas cidades do Rio de Janeiro e de São Paulo, as compras poderão ser efetuadas pessoalmente nos seguintes endereços: Rio de Janeiro - Rua Itapiru, 1209, Rio Comprido; São Paulo - Rua Frei Caneca, 1152, Consolação.

Distribuidor exclusivo para todo o Brasil: Fernando Chinaglia - Rua Teodoro da Silva, 907, Rio de Janeiro. Distribuidor para Portugal: Electroliber Lda. - Rua Prof. Reinaldo dos Santos, 1488, Lisboa.

INFORMAÇÃO EM CADEIA



Interface
Mesmo micros simples, como os ingleses da linha Sinclair, podem intercomunicar-se e partilhar recursos, quando equipados com interfaces apropriadas.

As redes não exigem equipamentos complicados. Compartilhando dados e periféricos caros, como impressoras, monta-se a custo compensador um sistema de microcomputadores interligados.

Há alguns requisitos a preencher antes de se conectarem computadores em rede. Eles devem estar próximos o suficiente para ser ligados por cabo — de preferência no mesmo prédio. É necessário, ainda, que haja tráfego de comunicação suficiente para que valha a pena montar a rede. Os usuários devem intercambiar dados muitas vezes por dia, ou partilhar um equipamento caro (tal como impressoras ou unidades de disco) para amortizar o custo da rede. Se apenas um pequeno número de dados passar pela rede, será mais fácil que cada usuário entregue ao outro uma fita ou um disco.

Cada computador tem seu próprio sistema operacional e seu próprio software para se comunicar com outros equipamentos. Conforme o tipo de configuração, todos os terminais de uma rede devem ser computadores do mesmo tipo. Por exemplo: todos IBM PC, ou todos Apple.

Vamos supor que um grupo de pessoas possua cinco computadores e queira compartilhar uma impressora. No caso, cada terminal deve enviar suas informações para a impressora.

O que fazer quando dois ou três terminais se dispõem a imprimir seus textos ao mesmo tempo? E se um terminal tem texto para imprimir mas quer também continuar trabalhando, enquanto a impressora opera?

Para resolver problemas como esses, instala-se um sexto computador, o controlador de impressão. Essa máquina tem como tarefa controlar o fluxo de dados a imprimir, sendo utilizada principalmente para esse fim. Ela armazena os documentos conforme a ordem de prioridade. Uma vez enviado o texto para o controlador de impressão, o terminal fica liberado para executar outras tarefas. É essencial que a rede use a máquina dedicada a controlar o fluxo das informações para a impressora, pois é por meio desse computador extra que as informações podem ser impressas.

Além do controlador de impressão, algumas aplicações de rede necessitam obrigatoriamente de um micro concentrador ou gerenciador de arquivos para lidar com unidades de disco compartilhadas e controlar o fluxo de informações entre os terminais.

O próximo passo: criar a ligação dos terminais. Para isso usa-se um cabo do tipo co-axial ou par trançado, que vai de máquina a máqui-

na. Embora haja várias maneiras de conectar os terminais e o micro concentrador, como a rede em estrela e a rede em anel, o conceito básico é o mesmo.

Montar a rede exige interface apropriada em todos os terminais. Essa interface pode ser uma simples conexão tipo RS232 ou uma placa que se encaixa num dos slots do micro. Além disso, o micro concentrador deve dispor de periféricos para armazenamento com capacidade suficiente para todo o fluxo de trabalho, assim como de RAM suficiente para gerenciar a rede.

O software utilizado determina a atuação da máquina e isso é válido sobretudo no caso das redes. Um sistema gerenciador da rede, acima do sistema operacional das máquinas, faz a conexão lógica de cada terminal com a rede. Conhecido como "protocolo", ele dá ao terminal concentrador o controle das operações de arquivo e/ou de impressão, e também do fluxo de dados pela rede.

Além de estabelecer essa cadeia de comandos, o protocolo informa a cada terminal que ele está em rede, subordinado a um micro gerenciador, e quantos outros terminais existem na rede. Por fim, o software de rede interage com o protocolo para que os terminais se comuniquem com o restante do sistema.

Experiência compartilhada
Esta escola foi equipada com micros dotados de monitores coloridos, uma unidade de disco rígido e uma impressora, o que fica mais barato do que equipar cada micro com uma unidade de disco e uma impressora em separado.
A velocidade da rede é tal que cada máquina parece acoplada a uma unidade de disco individual, mesmo quando trinta estudantes estão operando ao mesmo tempo. Às vezes os terminais ficam "na fila" quando querem utilizar a impressora. O sistema permite que o professor forneça experiência prática a todos os alunos. No ensino, a comunicação entre terminais é sempre uma grande vantagem.

Uma vez estabelecida a rede, cada terminal deve ter seu programa ou um conjunto de programas para suas próprias aplicações. Esses softwares reconhecem a rede e sabem como se comunicar com ela.

Desenvolvidos especialmente para aplicações em rede, tais programas podem ser executados a partir de um disco no próprio terminal ou por meio do micro concentrador.

Se o terminal estiver processando um texto, por exemplo, e enviando os resultados para a impressora, independentemente dos demais terminais, a única condição necessária ao processador de texto para que ele possa ser processado em rede é a inclusão de protocolos.

Por outro lado, se os terminais **B** e **D** tiverem de compartilhar os mesmos dados e um precisar consultar os resultados do outro, a operação em rede se complica. No caso, o software aplicativo (um processador de texto, uma planilha financeira, um banco de dados ou mesmo um jogo) e o hardware de sistema devem ser capazes de operar em conjunto.

Em outras palavras, a CPU (Unidade Central de Processamento) precisa executar várias tarefas ao mesmo tempo e manipular comunicações simultâneas provenientes de, pelo menos, duas outras CPUs.

Acesso e seleção

As redes locais têm inúmeras aplicações em pequenos e médios empreendimentos, comerciais e industriais. A uma administradora de imóveis, por exemplo, convém que todos os dados relativos a imóveis, condôminos, funcionários, despesas etc. fiquem registrados numa base de dados compartilhada. Essa base pode ser um disco rígido, caso o volume total de dados seja grande o bastante. O micro concentrador fica encarregado de lançar e atualizar os dados. Um segundo micro lança na mesma base os pagamentos em atraso, feitos na caixa da administradora. Outro emite os carnês de pagamento e os relatórios, e um quarto, utilizado pela contabilidade, controla pagamentos e débitos, envia cartas de cobrança e se encarrega da folha de pagamento dos funcionários.

Compartilhando uma impressora e um disco rígido, todos os serviços seriam executados com acesso a dados atualizados.

Por outro lado, pode-se evitar que os dados sejam indevidamente acessados estabelecendo restrições na configuração da rede. Por exemplo, o micro **A** acessa o **B** e o **C**, mas só é acessado pelo **D**; o **B** tem acesso ao **C** mas não ao **A**, e assim por diante. A própria base de dados pode ter setores restritos a este ou àquele micro. Dessa forma, todos trabalham com dados atualizados, mas só os acessam com prévia autorização.

Concentrador
Controla o tráfego da rede. Cada usuário tem sua própria senha para acessar a rede e seus próprios diretórios de arquivos. Há também um indice dos arquivos disponíveis a todos os usuários.

Caixa de sincronização
Gera os sinais de tempo que sincronizam todas as comunicações da rede.

EDITOR-CHEFE

Impressora
Acessada a partir de qualquer terminal da rede e controlada por ROM especial num dos terminais, deixando-o livre para o trabalho normal.

EDITOR DE PRODUÇÃO

Acoplador acústico (modem)
Passa informações por via telefônica da unidade de disco para um terminal remoto ligado à rede.

NÓ DE JUNÇÃO

SUBEDITOR

Terminador
Termina eletricamente as extremidades do cabo de comunicação (bus) para evitar a degradação dos sinais.

Identificação do terminal
Cada terminal tem um número específico (entre 001 e 254), fixado por chaves internas, para se identificar dentro da rede.

REDATOR

Acesso dos terminais
Todo terminal é capaz de enviar para qualquer outro, ou dele receber, mensagens que são mostradas na tela.

REDATOR

Edição em rede

Esta rede está sendo usada para produzir um jornal fictício. Os redatores utilizam terminais com processadores de texto e armazenam os textos na unidade de disco compartilhada. Sempre que desejar, a equipe de produção pode visualizar as telas dos redatores nas telas de seus próprios terminais e produzir o texto final a partir dos arquivos em disco. O micro do editor-chefe controla a rede, de modo que o texto final seja impresso. Arquivos de textos vindos de computadores remotos chegam ao concentrador por um modem.

DADOS NA TELA

Num jogo ocasional ou em cartas comerciais comuns, a qualidade da imagem é fator secundário. Mas, quando se destina à educação ou ao uso empresarial, crescem as vantagens do monitor de vídeo.

O vídeo, um dos periféricos mais importantes do computador, costuma ser o aparelho de TV comum, muitas vezes em preto e branco, comprado especialmente para esse fim. A qualidade assim obtida é aceitável, mas os resultados não são os melhores.

Isso porque o sinal enviado da memória de vídeo passa por vários estágios de codificação e decodificação antes de aparecer na tela. Por melhor que seja a qualidade do circuito em questão, o resultado é sempre imperfeito: as imagens finais muitas vezes saem borradas, difíceis de ler e com um desagradável efeito de tremulação.

O segredo da alta qualidade da imagem está na eliminação dessas distorções de sinal produzidas pelos circuitos de modulação e demodulação; para tanto, basta eliminar esses estágios.

É aí que se encontra a vantagem do monitor. Sem os estágios de codificação e decodificação utilizados na televisão, o monitor é um sistema simplificado capaz de gerar imagens mais nítidas, brilhantes e estáveis.

Como não existem decodificadores de sinal, o monitor não funcionará quando conectado à saída comum de televisão. O sistema exige saída de vídeo própria, que não passe pelo modulador. Para verificar isso, consulte o manual de seu computador.

O processo de gerar imagens na tela consiste, em grande parte, em assegurar que tudo aconteça no momento correto. O problema reside na produção da varredura do feixe luminoso dentro do monitor. Ou seja, ela não é controlada pelo computador.

Quando não está conectado a nenhum micro, o monitor produz um feixe que varre a tela sessenta vezes por segundo, formando um campo iluminado regular. Para transformar esse campo em imagem é necessário "ligar" e "desligar" o feixe exatamente no mesmo lugar a cada 1/60 de segundo. Qualquer instabilidade nesse processo resultará numa tremulação irritante, que, no melhor dos casos, torna a tela cansativa para o olhar.

Todo o processo depende dos "pulsos de sincronização" e do brilho produzidos pelo computador e enviados ao monitor.

Há dois tipos de pulso de sincronização: um para cada linha da imagem e outro para cada imagem completa. No final de cada ciclo, um pulso curto informa ao monitor que a imagem está completa, obrigando o feixe de elétrons (e, com isso, o ponto que ele produz) a retornar ao canto superior esquerdo da tela para repetir o ciclo.

O processo é o mesmo ao final de cada linha. O monitor recebe um pulso informando que a linha se completou, e o feixe de elétrons volta ao lado esquerdo da tela para iniciar a linha seguinte.

Há vários tipos de monitor, enquadrados em duas categorias principais — coloridos e monocromáticos — subdivididas nos diferentes tipos de sinal que podem receber.

PIXEL

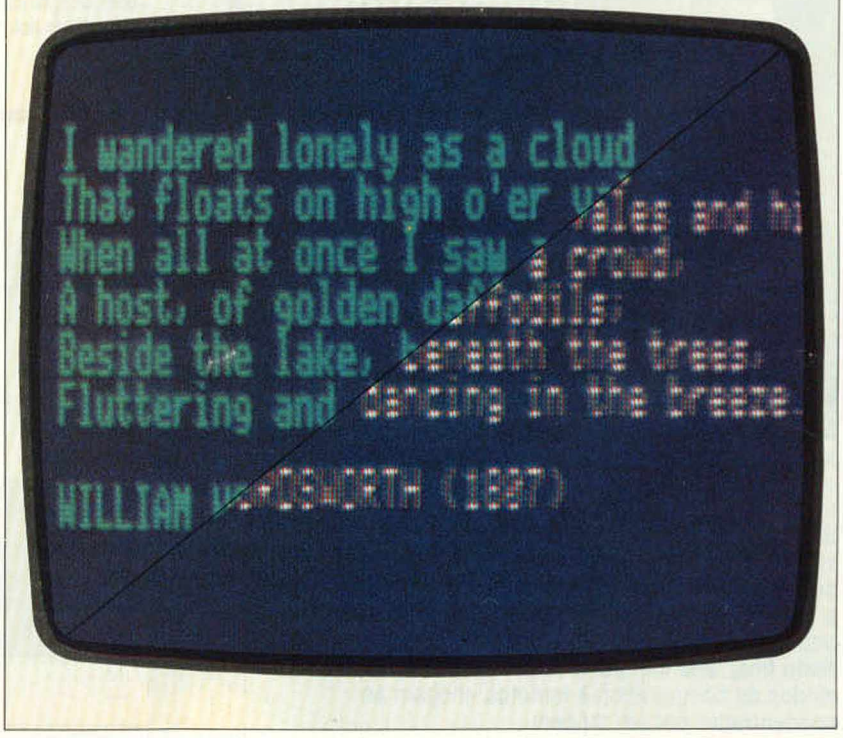
Palavra derivada de picture cell (célula da imagem), indica o menor conjunto de pontos da tela que pode ser acessado individualmente — ou seja: o bloco de construção das imagens.

Coloridos e monocromáticos

Os monitores monocromáticos mais utilizados são muito simples e trabalham com sinal composto: os pulsos de sincronização combinam-se com o nível de brilho num único sinal que o monitor utiliza para produzir a imagem.

Monitor x televisão

A tela mostra a grande diferença de qualidade de imagem obtida com monitor monocromático (no caso, o Monitor III, da Apple) e com aparelho de televisão comum, usados para apresentar a saída de um processador de texto.



Alguns monitores em cores funcionam com esquema semelhante, mas, devido à maior complexidade dos sinais, estão mais próximos dos aparelhos de televisão e operam em geral com os mesmos sistemas de codificação e decodificação.

Os tipos principais, PAL, SECAM e NTSC — nome dos sistemas usados na transmissão de televisão em diferentes partes do mundo —, representam métodos de codificar as cores básicas (vermelho, verde e azul) e os dois pulsos de sincronização.

Existe outro método, em que os vários sinais são enviados separadamente ao monitor. Embora haja variações sutis nos esquemas usados, estes são conhecidos como RGB, devido aos canhões de cores (Red, Green, Blue — vermelho, verde e azul). O mais simples é o TTL (transistor-transistor logic — lógica transistor-transistor), no qual as cores têm apenas dois estados: ligado e desligado. As oito cores usuais resultam das várias combinações dessas três.

Produz-se maior variedade de cores pela intensificação de cada cor básica. Embora a intensificação seja gradual, o resultado é conhecido como monitor "analógico" ou "linear". A maioria dos monitores em cores é encontrada tanto na versão TTL como na analógica, com pouca diferença de preço.

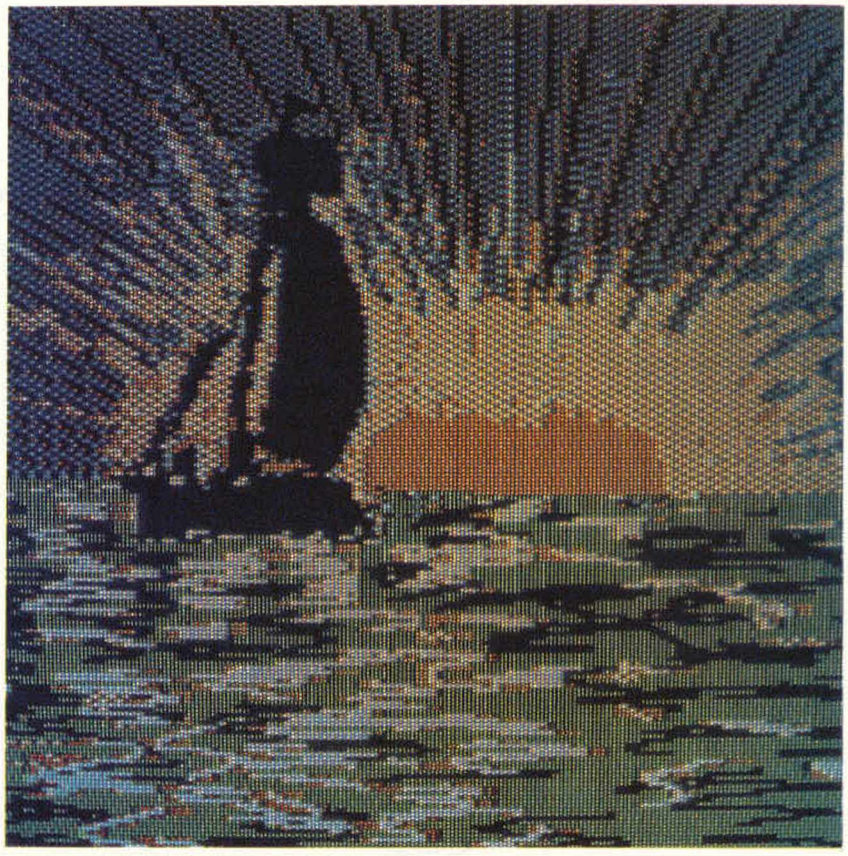
O TM90PSN, de fabricação japonesa, é um monitor interessante e adaptável. Possui um mecanismo seletor cuja função é verificar o sinal que está entrando e passar o monitor para o modo apropriado. Assim, o TM90PSN recebe qualquer tipo de entrada, vantagem que lhe valerá a popularização, pois serve como vídeo universal — para computador, videoteipe, videodisco e outros equipamentos.

Resolução e largura de banda

Na escolha do monitor, um dos aspectos a considerar é a quantidade total de pontos que o circuito pode acessar — isso define a resolução do aparelho. Essa quantidade depende do tamanho do pixel. De modo geral, quanto menor o pixel, melhor a resolução.

Para calcular a resolução, determina-se em primeiro lugar o número máximo de caracteres na tela — número de caracteres em cada linha vezes o número de linhas. Multiplicando o total obtido pelas dimensões da matriz de caracteres chega-se ao número total de pontos. A matriz é o total de pontos que cabe no retângulo da tela ocupado por um caractere. Numa tela comum de 80 colunas por 24 linhas, por exemplo, o total de caracteres é 80 x 24. Se a matriz for de 7 x 9, isto é, um retângulo com sete pontos por nove, a resolução será dada por (80 x 7) x (24 x 9), ou 560 x 216. O número máximo de pontos será o resultado, 120.960. Em geral, esse número está entre 10.000 e 1,5 milhão.

Os 120.960 pontos do exemplo são varridos pelo feixe eletrônico sessenta vezes por segundo — que é a freqüência da rede. Assim, o circuito de controle deve ligar e desligar o feixe luminoso 7.257.600 vezes por segundo (120.960 x 60).



Combinação de três
Nos monitores coloridos, ideais para a produção de gráficos e figuras, as oito cores usuais resultam da combinação das três cores básicas.
A intensificação das cores básicas, no entanto, permite aumentar a variedade de cores.

Esse número representa uma freqüência, chamada largura de banda, expressa em hertz (Hz). Por serem muito elevados, os valores das larguras de banda costumam ser indicados em milhões de hertz, ou megahertz (MHz). Existem monitores no mercado com larguras de banda de 5, 7, 10, 12, 15 e 20 MHz, e preços proporcionais a esses valores.

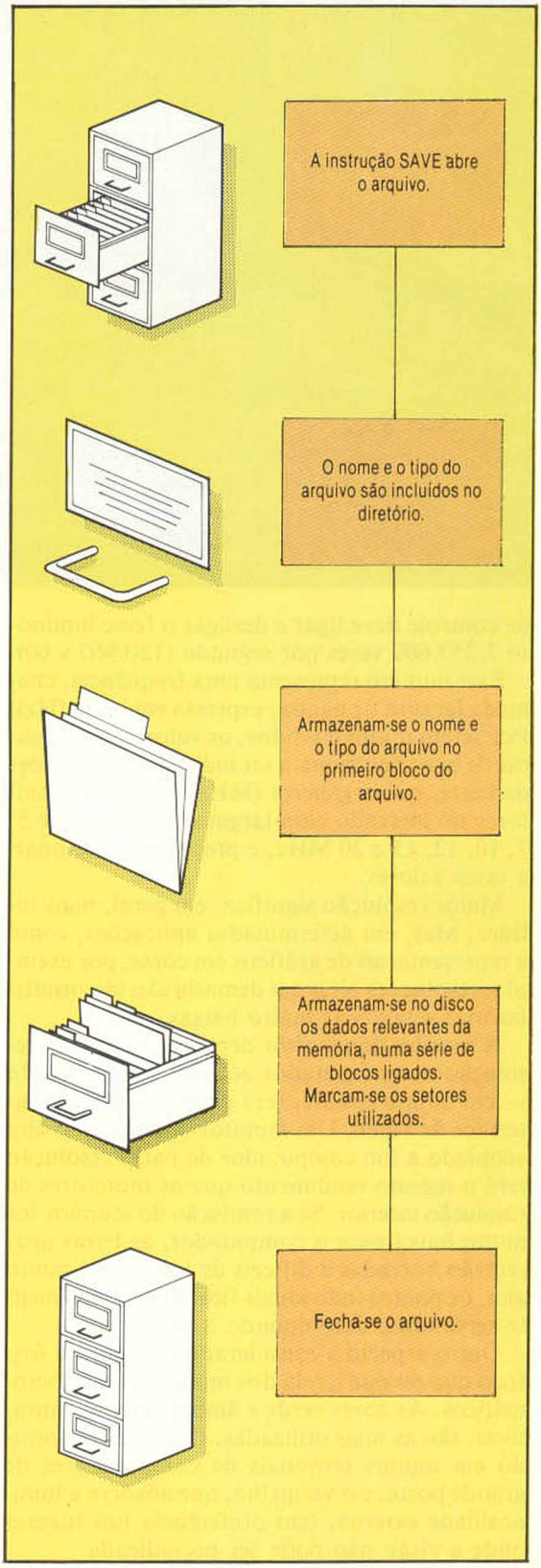
Maior resolução significa, em geral, mais nitidez. Mas, em determinadas aplicações, como a representação de gráficos em cores, por exemplo, resoluções altas em demasia são tão insatisfatórias quanto as muito baixas.

A escolha do monitor depende do tipo de resolução do computador ao qual será acoplado — em última análise, será determinada pela interface de vídeo. Um monitor de resolução alta acoplado a um computador de baixa resolução terá o mesmo rendimento que os monitores de resolução inferior. Se a resolução do monitor for muito baixa para o computador, as letras aparecerão borradas e difíceis de ler. Se for muito alta, os pontos individuais ficarão excessivamente separados, decompondo a imagem.

Outro aspecto a considerar é o da cor do fósforo que reveste a tela dos monitores monocromáticos. As cores verde e âmbar, menos cansativas, são as mais utilizadas. O azul é encontrado em muitos terminais de computadores de grande porte, e o vermelho, que absorve a luminosidade externa, tem preferência nos lugares onde a visão não pode ser prejudicada.

MEMÓRIA ORGANIZADA

Arquivamento direto
O arquivo binário copia parte da memória. Pode consistir num programa, numa imagem da tela etc. O arquivamento é direto. Depois de se entrar com o arquivo no diretório, os dados são gravados numa série de setores interligados. O DOS também mantém uma lista de setores usados, para evitar que sejam sobregravados por um novo arquivo.



Os arquivos — dados armazenados em computador — podem ser binários, seqüenciais ou aleatórios. E devem estruturar-se segundo o tipo, o volume e a velocidade das informações que concentram.

Arquivo é o termo mais adequado para designar o armazenamento de dados em computador. Sob vários aspectos, esse método se assemelha bastante ao de guardar documentos e revistas em pasta de arquivo.

O controle de nossas atividades diárias exige recursos como manter agenda de compromissos ou preencher o canhoto de cheque, fiscalizando o saldo bancário. Nessas situações, verifica-se a necessidade de guardar informações e de acessá-las com rapidez. Porém, à medida que aumenta o volume de informações manipuladas, torna-se mais complexo seu armazenamento.

Para gerenciar qualquer projeto ou empreendimento é básico ter em conta as constantes mudanças das informações em projetos que envolvem diferentes pessoas, lugares, objetos e circunstâncias. Muitos dos problemas enfrentados pelas empresas devem-se às dificuldades de controle e interpretação das informações. Portanto, a administração eficiente dos negócios exige a adoção de métodos simples de arquivamento, classificação e acesso aos dados.

Administradores competentes percebem a necessidade de estruturar o armazenamento segundo o tipo, o volume e a velocidade das informações sob seu controle. Esses princípios permanecem os mesmos quando se trata de sistemas de computação, projetados para manipular e armazenar — com precisão e em alta velocidade — grandes quantidades de dados.

No centro de tudo está o sistema operacional de disco (DOS, Disk Operating System) do computador, administrando as informações fornecidas pelo "gerente", isto é, seu programa. Assim, a eficiência dos sistemas de armazenamento depende do tipo de estruturação dos dados adotados pelo DOS. Também se deve levar em conta a forma de manipular esse sistema.

Os micros utilizam os métodos de manipulação de arquivo desenvolvidos, a princípio, para computadores de médio e grande porte: arquivos binários, seqüenciais e aleatórios.

Um bom exemplo de arquivo binário — simples cópia de parte da memória RAM disponível para o usuário — é o programa gravado.

Imagine a área de RAM como um bloco de anotações. Se você quiser guardar apontamen-

Diretório do disco

Na tela está um diretório de disco produzido pelo utilitário STAT, em CP/M, fornecendo muito mais informações do que a maioria dos diretórios.

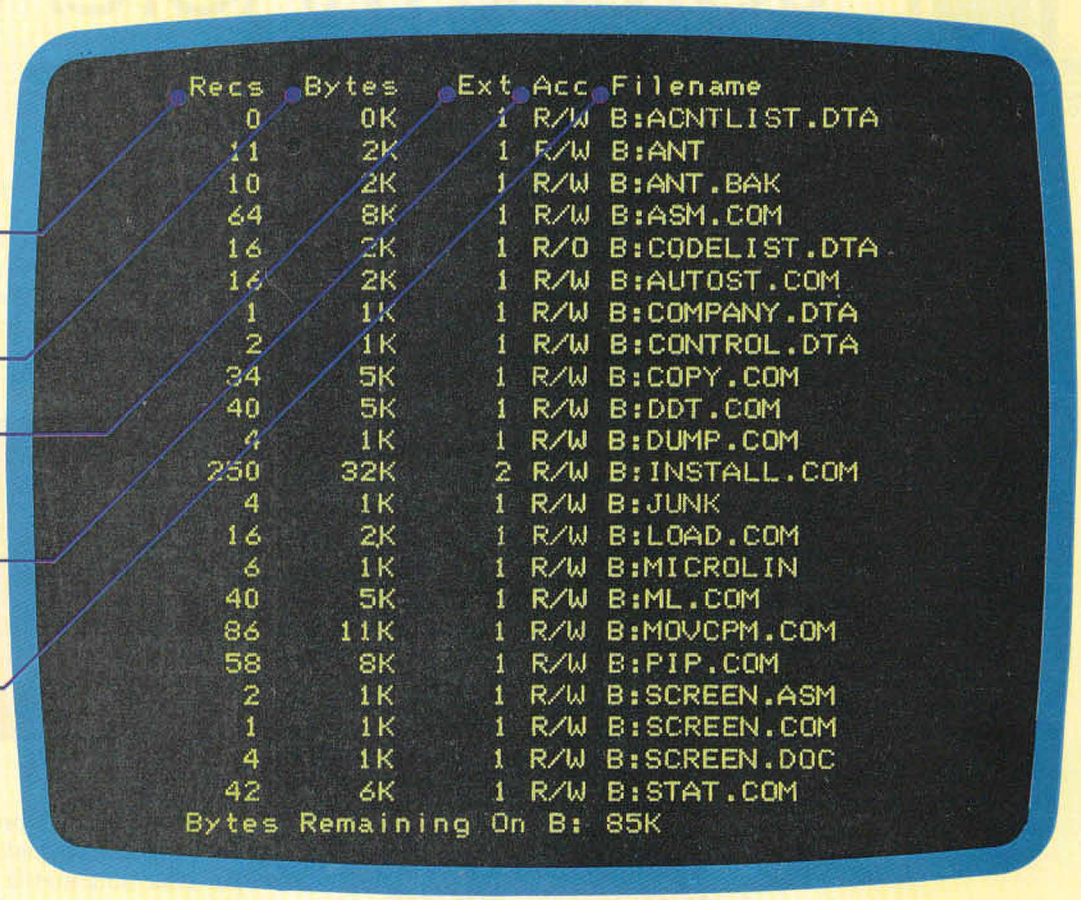
Recs
O número de registros (records) no arquivo pode variar. Aqui, eles têm 128 Kbytes de comprimento.

Bytes
O tamanho do arquivo é dado em Kbytes.

Ext
A extensão é uma medida alternativa do espaço em disco ocupado pelo arquivo.

Acc
Acesso: o arquivo pode ser marcado para leitura e gravação (R/W); ou apenas para leitura (R/O).

Nome do arquivo
O nome completo do arquivo se inicia com o nome da unidade de disco (A: ou B:), seguido pelo nome do arquivo (COPY, por exemplo) e, por último, pela extensão do arquivo (.COM, por exemplo), que fornece informações sobre o tipo de arquivo.



```
Recs  Bytes  Ext Acc Filename
   0     0K    1 R/W B:ACNTLIST.DTA
  11     2K    1 R/W B:ANT
  10     2K    1 R/W B:ANT.BAK
  64     8K    1 R/W B:ASM.COM
  16     2K    1 R/O B:CODELIST.DTA
  16     2K    1 R/W B:AUTOST.COM
   1     1K    1 R/W B:COMPANY.DTA
   2     1K    1 R/W B:CONTROL.DTA
  34     5K    1 R/W B:COPY.COM
  40     5K    1 R/W B:DDT.COM
   4     1K    1 R/W B:DUMP.COM
 250    32K    2 R/W B:INSTALL.COM
   4     1K    1 R/W B:JUNK
  16     2K    1 R/W B:LOAD.COM
   6     1K    1 R/W B:MICROLIN
  40     5K    1 R/W B:ML.COM
  86    11K    1 R/W B:MOVCPM.COM
  58     8K    1 R/W B:PIP.COM
   2     1K    1 R/W B:SCREEN.ASM
   1     1K    1 R/W B:SCREEN.COM
   4     1K    1 R/W B:SCREEN.DOC
  42     6K    1 R/W B:STAT.COM
Bytes Remaining On B: 85K
```

tos importantes ou desenhos interessantes, arranque as páginas em que eles estão registrados e guarde-as em lugar de fácil acesso.

Os arquivos binários funcionam da mesma forma. Ao receber um comando SAVE, o DOS registra no disco o nome do arquivo (FILENAME), marcando-o de forma especial como arquivo binário. Em seguida, a área relevante da memória é copiada byte a byte no disco. Armazena-se o programa em blocos interligados — com marcadores no final de cada bloco indicando o início do bloco seguinte — até chegar ao final do programa ou dos dados. O último bloco termina com um marcador de fim de arquivo.

Assim, demos nome a uma das páginas de nosso bloco de anotações e a guardamos numa gaveta de arquivo, além de termos incluído esse nome na lista de conteúdo da gaveta.

Ao receber um comando RETURN ou ENTER, cada linha de um programa em BASIC é codificada, com as palavras-chave da linguagem BASIC transformadas em números de 1 byte pelo interpretador BASIC. Isso porque os números são mais fáceis de manipular e para acessá-los basta sua decodificação em forma de texto.

Como a imagem da RAM, o arquivo binário também armazena códigos ASCII e dados binários. Gravar arquivos ASCII é útil para armazenar, por exemplo, o conteúdo da tela. Com esse recurso, os dados mostrados na tela são gravados e facilmente carregados na mesma área da memória.

Alguns DOS e algumas versões de BASIC permitem o armazenamento de programas em BASIC codificados em ASCII. Assim, sob a forma de um arquivo de texto pode-se editar o programa não codificado por meio de programas processadores de texto.

Apesar de muito fáceis de usar e controlar, os arquivos binários são limitados por dois fatores. Em primeiro lugar, só se gravam as informações em bloco contínuo de dados, o que obriga a recuperá-las da mesma forma. Para serem acessados, os arquivos binários devem estar integralmente carregados na memória. Além disso, o tamanho máximo de um arquivo é limitado pelo total de RAM disponível para o usuário.

Outras matérias desta série examinam os arquivos seqüenciais, que podem ser, dentro dos limites de espaço disponível no disco, tão grandes quanto necessário. Analisam também os arquivos de acesso aleatório, que permitem ao sistema operacional empregar diferentes métodos de arquivamento dos dados.

SOMAS MÁGICAS

O quadrado mágico — intrigante quebra-cabeça matemático — é um candidato ideal a experimentos. Sua implementação no microcomputador envolve alguns problemas interessantes.

O quadrado mágico é formado por linhas e colunas de números inteiros positivos, não repetidos, dispostos de modo que a soma de qualquer linha ou coluna tenha o mesmo resultado.

O zero não é permitido. Monta-se um quadrado mágico com qualquer quantidade de linhas, desde que igual à de colunas. Uma das soluções para o mais simples deles (ou seja, de 3 x 3) é esta:

7	6	2	15
3	8	4	15
5	1	9	15
15	15	15	

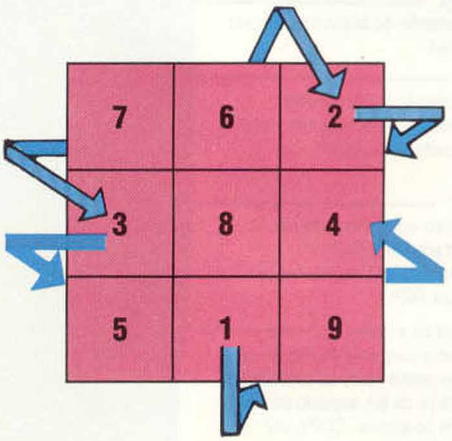
Nesse exemplo, a soma de qualquer linha ou coluna é 15. Como exercício, tente construir um quadrado mágico de 5 x 5 usando números de 1 a 25. Embora aparente ser fácil, esse quebra-cabeça exige diversas tentativas até alcançar a solução.

Utilizando o microcomputador, a tarefa fica bem mais simples. Bastaria, por exemplo, escrever um programa que introduzisse os números no quadrado e testasse todas as linhas e colunas. Encontrar a solução por esse método, no entanto, pode levar horas, mesmo para quadrados pequenos. O ideal é antes descobrir um processo para gerar os números. O método exposto a seguir sempre funciona e pode poupá-lo desse trabalho.

Comece colocando o número 1 na célula do meio da linha inferior.

Para acrescentar o número seguinte (no caso, o 2), mova uma célula para baixo e uma para a direita. Se esse movimento o levar para fora do quadrado, vá para a primeira linha de cima, conservando a mesma coluna. Se o deslocamento para a direita sair dos limites, ocupe a primeira coluna da esquerda, mantendo a posição da linha.

Caso a célula seguinte já esteja ocupada, coloque o número uma posição à esquerda da última célula usada e continue o processo até preencher todo o quadrado, como na ilustração.

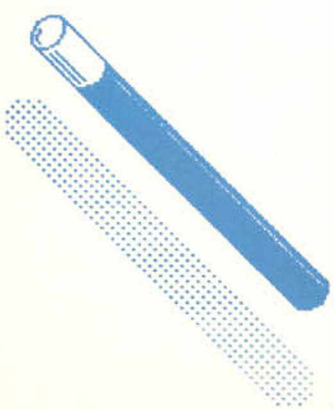


É fácil escrever um programa para gerar quadrados mágicos de qualquer tamanho. Começa-se por criar uma matriz quadrada vazia, do tamanho certo para o quadrado mágico desejado. Em seguida, recorre-se a um loop para preenchê-lo de acordo com as regras estabelecidas. Esse mé-

Quadrados de encantamento

Estes quadrados foram gerados e, depois, checados pelo programa. No quadrado de 15 x 15, observa-se uma seqüência de números de dois e três dígitos, que resulta do algoritmo usado em sua construção.

52	42	32	22	12	2	73	72	62
63	53	43	33	23	13	3	74	64
65	55	54	44	34	24	14	4	75
76	66	56	46	45	35	25	15	5
6	77	67	57	47	37	36	26	16
17	7	78	68	58	48	38	28	27
19	18	8	79	69	59	49	39	29
30	20	10	9	80	70	60	50	40
41	31	21	11	1	81	71	61	51



todo só funciona para quadrados mágicos com número ímpar de células (3 x 3, 5 x 5, 7 x 7, ...). Assim, o programa deve rejeitar matrizes com número par de células.

A clareza dos números na tela é o meio mais seguro de elaborar o quadrado. O alinhamento correto dos números em fileiras e colunas será simples se seu micro aceitar o comando PRINT USING. Se não, converta o número a ser impresso num string alfanumérico, usando caracteres de espaço, a fim de que todos os números tenham o mesmo comprimento. Para tanto, utilize a sub-rotina a seguir, que se adapta com facilidade a qualquer micro.

```
1000 REM Converte A em A$ e alinha
1010 A$ = STR(A)
1020 IF LEN (A$) < 3 THEN A$ = " " + A$: GOTO
1030 RETURN
```

O tamanho da tela costuma restringir a produção de grandes quadrados mágicos. As telas de 40 colunas comportam 13 colunas de dois dígitos mais as colunas de espaço, mas os quadrados de 13 x 13 incluem números de três algarismos. Nessas condições, o maior quadrado que cabe na tela é o de 9 x 9. Usando uma impressora, geram-se quadrados muito maiores. Em geral, as impressoras têm largura máxima de 132 colunas; quando se desejam quadrados maiores, podem-se imprimi-los em partes que serão unidas depois.

O objetivo principal desse projeto é criar o maior quadrado mágico compatível com a tela e apresentá-lo da forma mais clara. Faça seu programa e lembre-se de incluir comentários explicativos e uma rotina de verificação dos dados da entrada quanto ao tamanho pretendido.

130	114	98	82	66	50	34	18	2	211	210	194	178	162	146
147	131	115	99	83	67	51	35	19	3	212	196	195	179	163
164	148	132	116	100	84	68	52	36	20	4	213	197	181	180
166	165	149	133	117	101	85	69	53	37	21	5	214	198	182
183	167	151	150	134	118	102	86	70	54	38	22	6	215	199
200	184	168	152	136	135	119	103	87	71	55	39	23	7	216
217	201	185	169	153	137	121	120	104	88	72	56	40	24	8
9	218	202	186	170	154	138	122	106	105	89	73	57	41	25
26	10	219	203	187	171	155	139	123	107	91	90	74	58	42
43	27	11	220	204	188	172	156	140	124	108	92	76	75	59
60	44	28	12	221	205	189	173	157	141	125	109	93	77	61
62	46	45	29	13	222	206	190	174	158	142	126	110	94	78
79	63	47	31	30	14	223	207	191	175	159	143	127	111	95
96	80	64	48	32	16	15	224	208	192	176	160	144	128	112
113	97	81	65	49	33	17	1	225	209	193	177	161	145	129

```
10  REM ************************
20  REM *   QUADRADOS  MAGICOS   *
30  REM ************************
40  M=19:DIM A(M,M)
50  PRINT:PRINT "QUADRADOS MAGICOS"
60  PRINT:PRINT "QUANTAS LINHAS (1 A 19)";:
    INPUT S
70  IF S<0 OR S<>INT(S) THEN PRINT "ERRO":
    GOTO 60
80  IF S>M THEN PRINT "ERRO":GOTO 60
90  IF S/2=INT(S/2) THEN PRINT "ERRO, APENAS
    NUMERO IMPAR":GOTO 60
100 REM *** GERA QUADRADO ***
110 X=INT(S/2)+1:Y=S:C=1
120 A(X,Y)=C:C=C+1:IF C>S*S THEN GOTO 200
130 X=X+1:IF X>S THEN X=1
140 Y=Y+1:IF Y>S THEN Y=1
150 IF A(X,Y)<>0 THEN X=X-2:Y=Y-1
160 IF Y=0 THEN Y=S
170 IF X=0 THEN X=S
180 IF X=-1 THEN X=S-1
190 GOTO 120
200 REM *** MOSTRA QUADRADO ***
210 PRINT:PRINT
220 FOR Y=1 TO S:FOR X=1 TO S
230 A=A(X,Y):GOSUB 390  :PRINT A$;
240 NEXT X:PRINT:NEXT Y
250 REM *** VERIFICA LINHAS E COLUNAS ***
260 F=0
270 FOR Y=1 TO S:T=0
280 FOR X=1 TO S:T=T+A(X,Y):NEXT X
290 IF F=0 THEN U=T:F=1
300 IF T<>U THEN PRINT "ERRO - LINHA 1 E
    LINHA ";Y;" NAO COMBINAM":END
310 U=T:NEXT Y
320 FOR X=1 TO S:T=0
330 FOR Y=1 TO S:T=T+A(X,Y):NEXT Y
340 IF T<>U THEN PRINT "ERRO - LINHA 1 E
    COLUNA ";X;" NAO COMBINAM":END
350 U=T:NEXT X
360 PRINT "TODAS AS LINHAS E COLUNAS
    ADICIONADAS SOMAM ";T;
370 GET A$:IF A$="" THEN GOTO 370
380 END
390 REM *** CONVERTE NUMERO EM STRING ***
400 A$=STR$(A)
410 IF LEN(A$)<4 THEN A$=" "+A$:GOTO 410
420 RETURN
```

Variações

Embora escrito para a família Apple, este programa pode ser rodado em quase todos os micros. Para o CP 400 e o CP 500, troque apenas a instrução GET A$ da linha 370 por A$ = INKEY$. Devido à limitação da tela, é possível fazer quadrados de 7 x 7 no CP 400, 13 x 13 no CP 500, 9 x 9 no Apple de 40 colunas e 19 x 19 no Apple de 80 colunas.

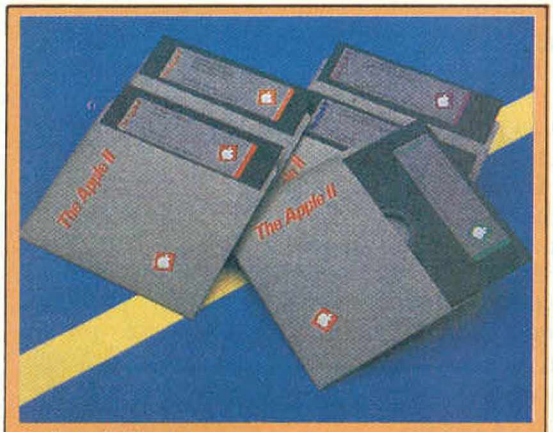
Para a linha Sinclair, insira LET antes de todas as atribuições de valor. O programa pede o número de linhas (e, portanto, o de colunas) da matriz e verifica se é inteiro, positivo e ímpar. Daí, gera e exibe o quadrado mágico na tela. A partir da linha 250, checa os resultados. Se isso lhe parecer desnecessário, omita as linhas 250 e 360.

APPLE IIc

Sem compatíveis no Brasil, o Apple IIc (o *c* significa "compacto"), lançamento recente da família Apple, apresenta avanços em relação aos modelos II, II+ e IIe, mas mantém compatibilidade com eles.

O Apple IIc é um modelo portátil, com a metade da altura de seus antecessores, os Apple II, II+ e IIe, e com uma unidade de disco embutida no lado direito. Pesa apenas 3,4 kg e já vem com uma alça embutida, que se transforma em apoio para colocar a máquina em ângulo confortável para o usuário.

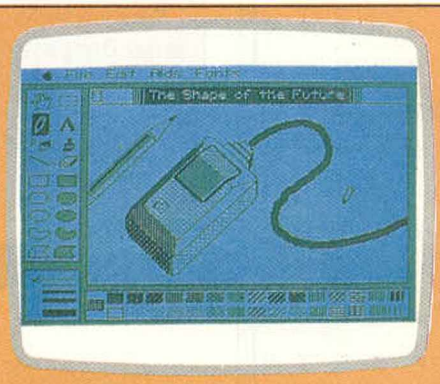
Ao contrário das outras máquinas da família Apple, o IIc é uma máquina com arquitetura fechada, sem slots (encaixes) para o acoplamento de expansões e periféricos. Em vez disso, já é dotado internamente da maioria das interfaces e das conexões que antes necessitariam de placas especiais. Há saídas para impressora, para um segundo drive, para modem, para TV doméstica, vídeo composto e RGB, e uma saída de nove pinos que serve para joystick ou mouse. O IIc já vem com 128 Kbytes de memória RAM e 80 colunas de vídeo. Sua tela também foi consideravelmente melhorada. Além das duas opções no modo texto (40 ou 80 colunas), no modo gráfico ela apresenta dezesseis cores e três tipos de resolução: baixa, alta e superalta. No começo de 1985, as indústrias Apple planejavam o lançamento de um visor de cristal líquido (LCD) de 24 por 80 colunas. Com isso e com o sistema de alimentação a pilha também em estudo, o Apple IIc se tornará totalmente portátil.



Conjunto de disquetes
O Apple IIc vem com um conjunto de cinco disquetes, três dos quais com noções de BASIC, LOGO e instruções sobre a máquina. O quarto é um software integrado que combina planilha financeira e banco de dados. E, por fim, um disquete que contém o sistema operacional PRODOS.

Emprega o sistema operacional DOS 3.3 ou o novo PRODOS, que trabalha com armazenamento dos arquivos em árvore. Cada arquivo nesse sistema é incluído numa estrutura hierárquica, facilitando a organização e o acesso dos arquivos no disco. Inteiramente compatível com os outros Apple II, o modelo compacto dispõe de imensa quantidade de softwares: mais de 17.000 programas, de todos os tipos, foram escritos para o Apple II. Além destes, para o próprio IIc desenvolveu-se o Appleworks, um software que combina processador de texto, banco de dados e planilha financeira. Janelas na tela facilitam a total integração das três aplicações. Dispõe também do MousePaint, um gerador de imagens que vem com o mouse opcional.

Acompanham o Apple IIc cinco discos: um com os programas utilitários do sistema operacional PRODOS e quatro com demonstrações a respeito do próprio equipamento, o software Apple-works, o BASIC e o LOGO.



MousePaint
Este software gráfico acompanha o mouse, disponível para os modelos II+, IIe e IIc. É uma versão simplificada do MacPaint da Macintosh e permite criar desenhos na tela com muita facilidade. Um menu apresenta opções sob a forma de figuras. O MousePaint aproveita as cores e a alta resolução do IIc. O mouse não exige interface especial para se acoplar ao Apple IIc.

Saída padrão RS232 para impressora

Saída para vídeo composto

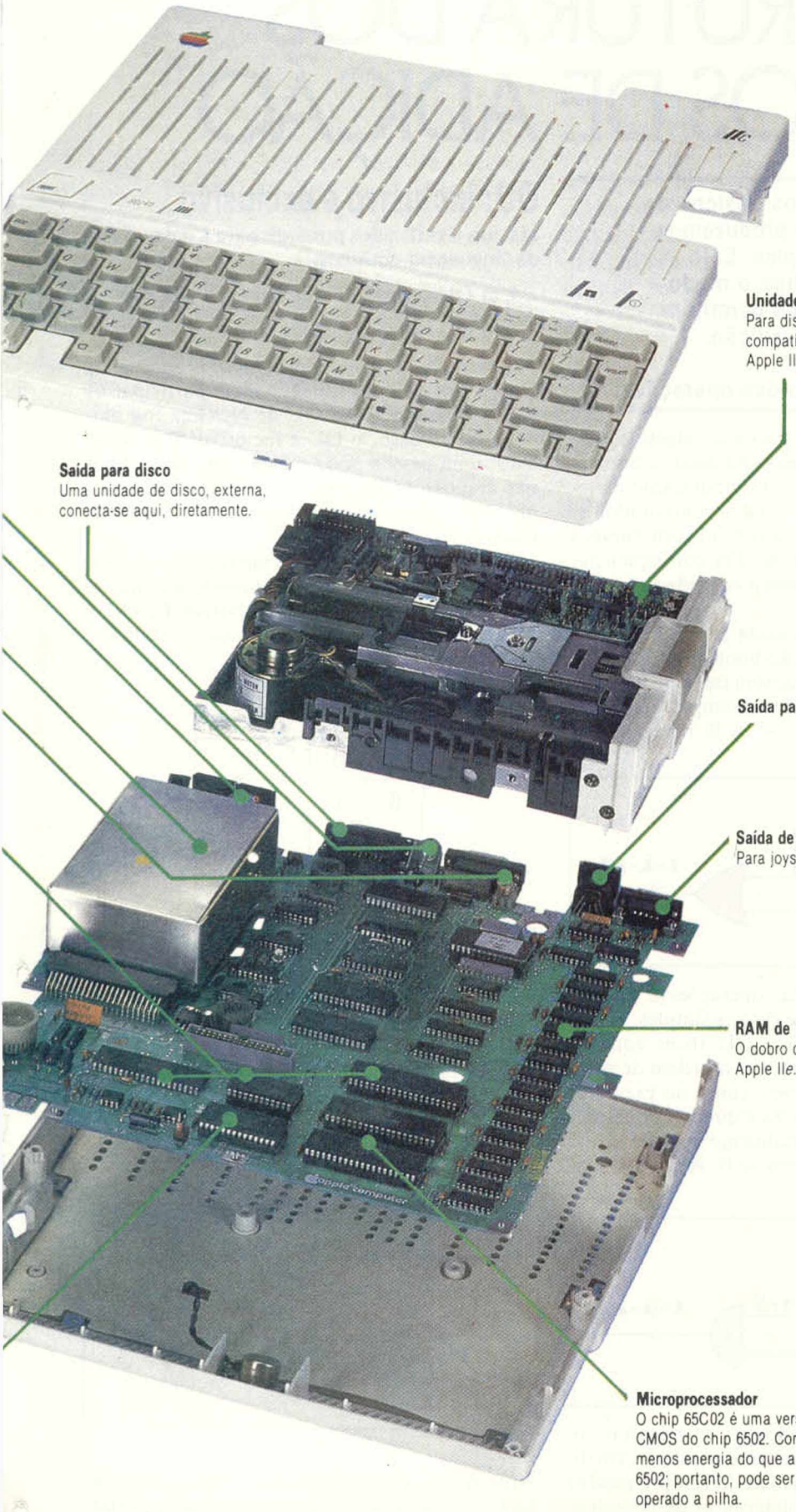
Fonte de alimentação
O IIc tem uma fonte interna (bateria) de 12 volts e requer um transformador para a corrente de 110/220 VAC.

Saída para vídeo RGB
Um pequeno adaptador PAL conectado a esta saída permite que o IIc se acople a uma TV comum.

Controladores de entrada e saída (I/O)
Estes chips controlam o teclado e todas as saídas para periféricos.

Saída de áudio
Conexão para amplificador de som externo.

ROM
Com 16 K, contém as rotinas de processamento interno e o BASIC Applesoft.



Unidade de disco interna
Para discos de 143 K, compatível com o Apple II+ e o Apple IIe.

Saída para disco
Uma unidade de disco, externa, conecta-se aqui, diretamente.

Saída para modem

Saída de nove pinos
Para joystick ou mouse.

RAM de 128K
O dobro da RAM padrão do Apple IIe.

Microprocessador
O chip 65C02 é uma versão CMOS do chip 6502. Consome menos energia do que a versão 6502; portanto, pode ser operado a pilha.

APPLE IIc

MICROPROCESSADOR
65C02

CLOCK
1 MHz

MEMÓRIA
16 Kbytes de ROM
e 128 Kbytes de RAM

SISTEMA OPERACIONAL
DOS 3.3 ou PRODOS

VÍDEO
Monitor de fósforo verde opcional.
Modo texto: 24 linhas de 40 ou 80 caracteres.
Modo gráfico: três níveis de resolução: baixa (40 x 40 pixels), alta (240 x 192) e superalta (560 x 192), com dezesseis cores.

TECLADO
Profissional, com 63 teclas, sendo quatro de movimentação do cursor. Dispõe de uma chave para alternar entre 40 e 80 caracteres e de outra para transformar o teclado QWERTY ou DVORAK.

LINGUAGENS
Residentes: ASSEMBLER e BASIC (versão Applesoft). Disponíveis: LOGO, PASCAL, FORTRAN.

PERIFÉRICOS
Conexões internas para impressora, unidade de disco, modem, joystick, mouse, amplificador de som, TV doméstica, vídeo composto e RGB.

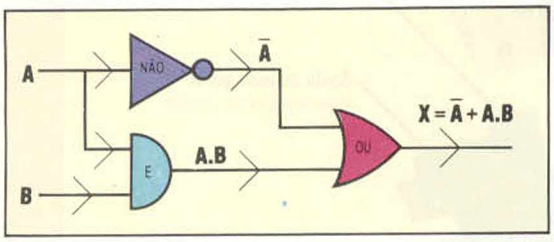
DOCUMENTAÇÃO
Ótimo guia do usuário, bem ilustrado, e um disco de demonstração interativa, *Apple presents Apple*.

A ESTRUTURA DOS BLOCOS DE ADIÇÃO

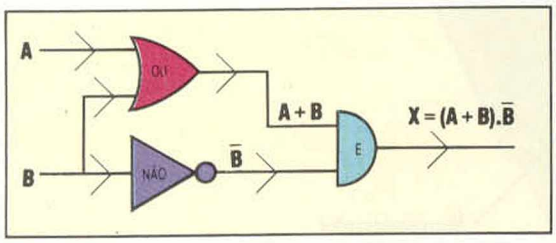
Com os três elementos básicos da lógica (E, OU e NÃO) produzem-se circuitos lógicos simples. Esta parte do curso investiga o modo pelo qual tais circuitos permitem executar a função da adição, considerando ainda a ordem da execução das operações.

O sistema que emprega a notação algébrica para descrever relações lógicas é a álgebra booleana. A esse sistema deve-se a simplificação matemática dos circuitos lógicos dos computadores, que permite executar uma função com número reduzido de portas lógicas. Em conseqüência, aumenta consideravelmente a velocidade de operação da máquina.

As expressões para a saída a partir das três portas lógicas (em notação booleana), E ($A \cdot B$), OU ($A + B$) e NÃO ($\bar{A}$), servem também para circuitos mais complexos. Por exemplo, a expressão booleana $X = \bar{A} + A \cdot B$ representa o seguinte circuito:



A ordem de execução das operações E e OU é relevante. A regra, nesse caso, é simples: E tem prioridade com relação a OU (bem como a NÃO). Se for preciso inverter a ordem de prioridade, usam-se parênteses, como no exemplo: $X = (A + B) \cdot \bar{B}$. Para essa expressão, operam-se as duas entradas inicialmente em OU e, depois, em E, com a negativa de B. Abaixo, o diagrama do circuito:



Ao desenhar um circuito lógico, convém que você comece pela saída e trabalhe para trás, em direção às entradas. Esse método sempre produz melhores esquemas de diagramas de circuitos.

OU: inclusivo e exclusivo

Há dois significados possíveis para a palavra OU na linguagem cotidiana:

Um OU outro OU ambos.

Um OU outro, mas não ambos.

Por exemplo, numa competição de proprietários de veículos de duas rodas, podem participar os donos de motocicleta OU de bicicleta (ou ambos). Nesse caso, o OU é inclusivo. Por outro lado, uma pessoa pode ser alta OU baixa (mas não ambos). O uso da palavra OU, nesse exemplo, exclui a possibilidade de que as duas afirmações sejam verdadeiras.

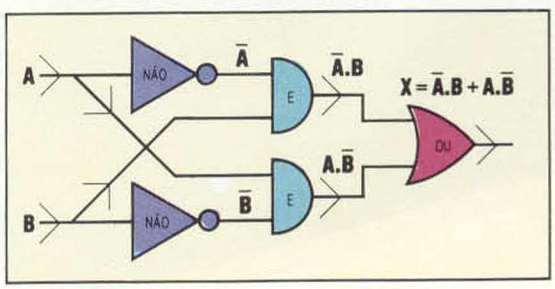
Nos circuitos lógicos, essa operação do OU exclusivo (ou XOR) é um instrumento útil, que se forma com um conjunto de portas E, OU e NÃO. A tabela de validação para o XOR é:

ENTRADAS		SAÍDAS
A	B	$A \veebar B$
0	0	0
0	1	1
1	0	1
1	1	0

Pela análise da segunda e da terceira linhas da tabela de validação, chega-se à conclusão de que a saída será 1 se:

NÃO (A) for operado em E com B
OU
A for operado em E com NÃO (B).

Essas duas possibilidades são representadas pela expressão booleana: $X = \bar{A} \cdot B + A \cdot \bar{B}$, da qual resulta um circuito de cinco portas capaz de produzir operação OU exclusivo:



Aplicando a álgebra booleana é possível simplificar esse circuito tornando-o de quatro portas.

Aritmética e portas lógicas

Embora a maioria dos micros execute toda a série de funções aritméticas, apenas a adição cabe aos circuitos lógicos. Realizam-se as demais, como subtração, multiplicação e divisão, por meio de uma combinação de software e de circuitos "somadores" do hardware para controlar o movimento das configurações de bits.

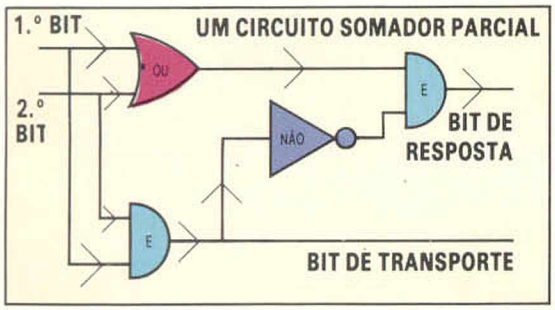
O conhecimento dos circuitos necessários para efetuar a adição binária exige que antes se examine o processo em si. Assim, considere, por exemplo, a adição binária:

	8s	4s	2s	1s
	0	1	0	1
	0	1	1	1
	1	1	0	0
TRANSPORTES	1	1	1	

Tomando uma coluna da soma isoladamente — por exemplo, a coluna dos dois — é possível relacionar suas várias entradas e saídas. *Entradas:* os dois bits a serem somados e o bit de transporte da coluna anterior. *Saídas:* o bit colocado na coluna dos dois como resposta e o bit a ser transportado para a coluna seguinte. O recebimento dessas entradas e a produção da saída correta cabem ao somador. O funcionamento desse dispositivo pode ser compreendido mediante uma versão simplificada, o somador parcial, que não considera a possibilidade de um transporte da coluna anterior. Isso reduz o problema a um circuito com duas entradas e duas saídas. E daí resulta uma tabela de validação para o circuito de um somador parcial:

ENTRADAS		SAÍDAS	
1.° BIT	2.° BIT	BIT DE TRANSPORTE	BIT DE RESPOSTA
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Pela tabela, percebe-se que o bit de transporte será 1 se o primeiro bit E o segundo forem ambos 1. O bit de resposta é formado pela operação em OU das entradas, desde que o bit de transporte não seja igual a 1. A saída do bit de resposta será 1 se o primeiro bit for 1 OU se o segundo for 1, e NÃO se o bit de transporte for 1. O circuito lógico abaixo produzirá as saídas desejadas:



EXERCÍCIO 2

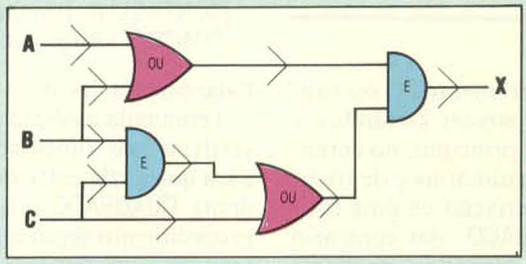
1) Desenhe os circuitos lógicos para as expressões booleanas:

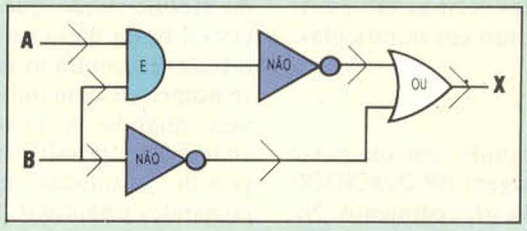
a) $X = (A + B)\cdot C$
b) $X = A\cdot B + (A + C)$
c) $X = \bar{A}\cdot B + (A + B)$
d) $X = A\cdot\bar{B}\cdot(A + B)$

2) Escreva a expressão booleana usando A e B como entradas para o bit de transporte e para o bit de resposta de um circuito somador parcial.

3) Escreva as expressões booleanas para os circuitos:

Respostas na próxima matéria desta seção.





Respostas do exercício anterior

1)

PASSOU NO EXAME DE HABILITAÇÃO	ACOMPANHADO POR UM MOTORISTA QUALIFICADO	PODE DIRIGIR UM CARRO
FALSO	FALSO	FALSO
FALSO	VERDADEIRO	VERDADEIRO
VERDADEIRO	FALSO	VERDADEIRO
VERDADEIRO	VERDADEIRO	VERDADEIRO

2)

GRAVADOR CASSETE DISPONÍVEL	UNIDADE DE DISCOS DISPONÍVEL	DESENVOLVIDO PARA COMPUTADOR DIFERENTE	O PROGRAMA SERÁ CARREGADO
FALSO	FALSO	FALSO	FALSO
FALSO	FALSO	VERDADEIRO	FALSO
FALSO	VERDADEIRO	FALSO	VERDADEIRO
FALSO	VERDADEIRO	VERDADEIRO	FALSO
VERDADEIRO	FALSO	FALSO	VERDADEIRO
VERDADEIRO	FALSO	VERDADEIRO	FALSO
VERDADEIRO	VERDADEIRO	FALSO	VERDADEIRO
VERDADEIRO	VERDADEIRO	VERDADEIRO	FALSO

3)

A	B	P	Q	C
0	0	1	0	0
0	1	1	1	1
1	0	0	1	0
1	1	0	1	0

A ARTE DA TARTARUGA

Na linguagem LOGO, além de operar no modo imediato, o programador pode definir como rotina uma seqüência de comandos sob nomes específicos. Para carregá-los, basta escrever o nome.

À primeira vista, os "procedimentos", ou sub-rotinas, do LOGO podem parecer estranhos a muitos programadores. Seu princípio, no entanto, é o mesmo das receitas culinárias e de tricô. Este capítulo começa pela criação de uma rotina LOGO, chamada QUADRADO. Ao contrário dos comandos — sempre digitados em letras maiúsculas —, os nomes das rotinas são escritos tanto em maiúsculas como em minúsculas.

De início digita-se:

```
EDITE QUADRADO
```

A tela ficará limpa. Em seguida, em sua parte superior aparecerá a mensagem AP QUADRADO — um lembrete do nome do procedimento. Na parte inferior da tela, surgirá a mensagem: EDICAO: CTRL-C = FIM, CTRL-G = CANCELAR. Um tanto enigmática, essa mensagem serve para ajudá-lo a operar em modo de edição. Isso significa apenas que o LOGO não está operando em modo imediato, mas esperando uma rotina. Após digitar a rotina, pressione as teclas [CONTROL]-C para registrá-la. Se for necessário interromper e começar de novo, pressione [CONTROL]-G.

Quando o LOGO estiver em modo EDITE, nada do que se digitar será executado. O LOGO apenas recebe e armazena a seqüência de instruções em seu "dicionário" sob o nome dado à rotina. Complete a definição do procedimento QUADRADO, introduzindo os comandos:

```
AP QUADRADO
REPITA 4 [FR 50 DI 90]
FIM
```

Para fazer com que o LOGO registre esse procedimento, pressione CONTROL-C. Na tela surgirá a mensagem QUADRADO OK. Se houver algum erro, interrompa o processo, pressionando o CONTROL-G. Nesse caso, refaz-se o procedimento desde o início, ou usam-se os comandos de edição em modo EDITE para introduzir as correções. Em modo imediato, o editor de linhas permite apenas correções na linha que acabou de ser digitada. Digitando EDITE, seguido de um espaço e o nome do procedimento (por exemplo, EDITE QUADRADO), e depois teclando RETURN, aparecerá no alto da tela o nome do procedimento com o cursor logo abaixo. Isso permite posicionar o cursor a fim de corrigir o que for necessário. Além de mover-se com as setas, o cursor obedece aos comandos:

```
[ESC] elimina o caractere à esquerda do cursor
[CONTROL]-D elimina o caractere sob o cursor
[CONTROL]-N avança uma linha
[CONTROL]-P retrocede uma linha
```

Falaremos deles mais adiante.

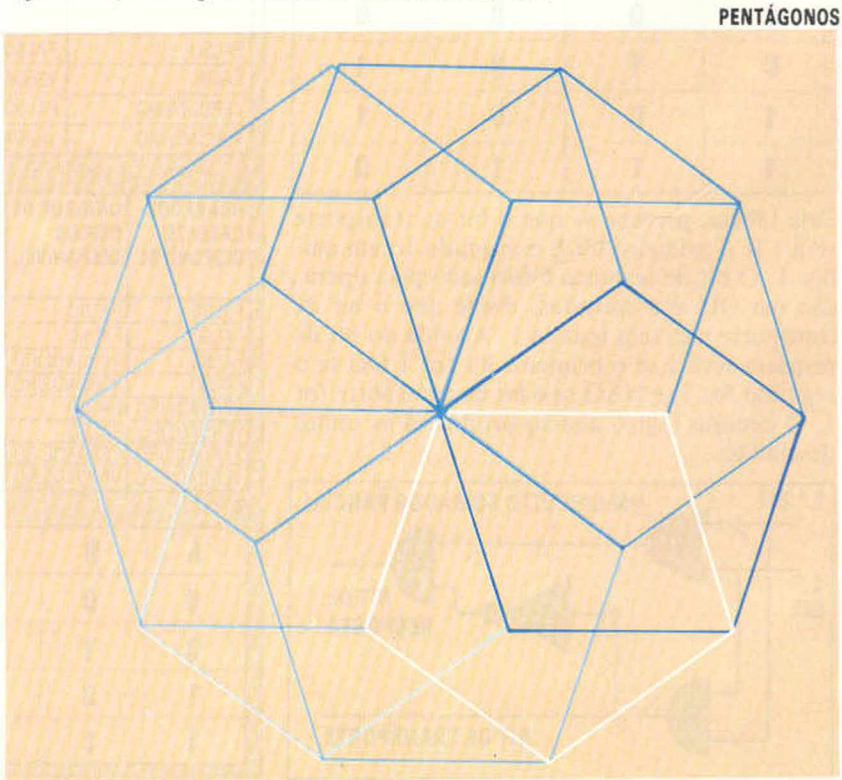
Terminada a edição e gravação da rotina, resta verificar seu funcionamento. Digite DESENHE para que a tela entre no modo gráfico. A seguir, digite QUADRADO. Os comandos definidos no procedimento serão executados e a tartaruga desenhará o quadrado. Usam-se os procedimentos do mesmo modo que os comandos básicos do LOGO: basta digitar o nome do procedimento e o LOGO executará as instruções definidas sob esse nome. Os comandos originais — já disponíveis quando o LOGO é carregado — são chamados "primitivos". As novas rotinas, depois de "ensinadas" no LOGO, são usadas como comandos primitivos. O LOGO é, assim, uma linguagem extensível, capaz de armazenar uma biblioteca de comandos que atenda a necessidades específicas.

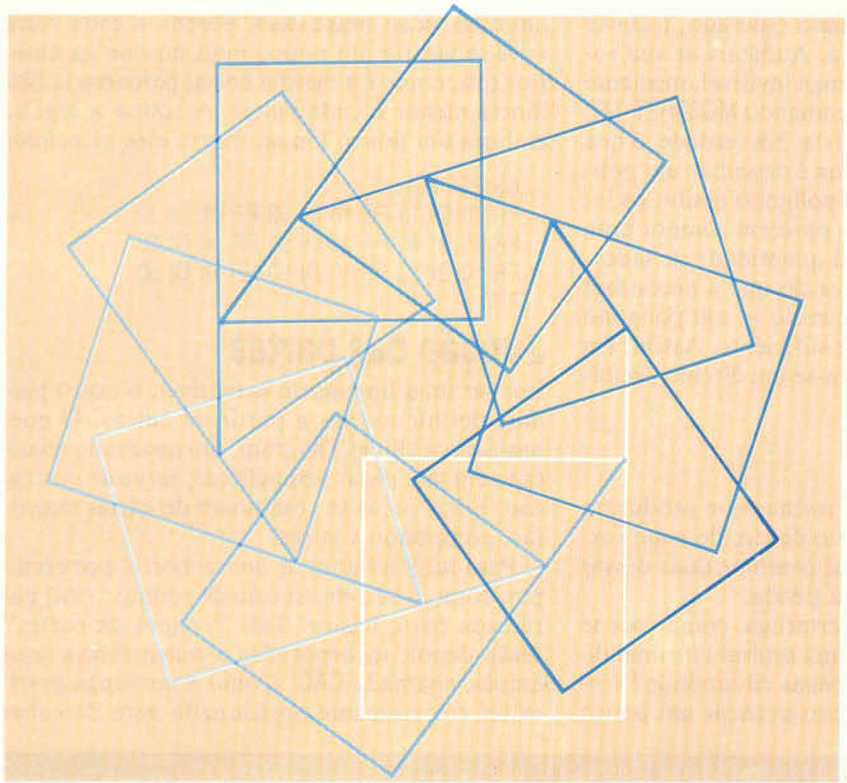
Quando uma rotina não produz os resultados esperados, é simples modificá-la. A rotina do

Simplificação

EDITE	ED
SEMT	ST
MOSTRET	MT
LISTAR	LI

PENTÁGONOS





QUADRADOS

exemplo dado traça um quadrado cujos lados medem 50 unidades. Para obter um quadrado com lados de apenas 30 unidades, retorne ao modo EDITE, digitando:

```
ED QUADRADO
```

O texto da rotina aparece na tela. Utilizando o editor de tela, substitua 50 por 30. Em seguida, grave o novo procedimento, pressionando CONTROL-C. A seguir, digite QUADRADO e observe se o tamanho do quadrado é o desejado.

Como exercício, tente desenvolver procedimentos que produzam figuras complexas a partir das formas criadas no capítulo anterior: triângulos, retângulos, pentágonos, estrelas etc. Utilize o comando REPITA com esses procedimentos. Por exemplo, para o pentágono, defina o procedimento PENT como:

```
REPITA 5 [FR 50 DI 72]
```

A seguir, experimente:

```
REPITA 10 [PENT DI 36]
```

O comando REPITA permite traçar figuras complexas com rapidez e facilidade. Tente criar uma figura a partir de quadrados:

```
REPITA 10 [QUADRADO DI 36 FR 25]
```

As ilustrações mostram os resultados desses procedimentos.

Respostas do exercício 1

Triângulo:
REPITA 3 [FR 50 DI 120]

Pentágono:
REPITA 5 [FR 50 DI 72]

Hexágono:
REPITA 6 [FR 60 DI 60]

Retângulo:
REPITA 2 [FR 25 DI 90 FR 50 DI 90]

Estrela de cinco pontas:
REPITA 5 [FR 50 DI 144]

Estrela de dez pontas:
REPITA 10 [FR 50 DI 108]

Paralelogramo:
REPITA 2 [FR 25 DI 70 FR 50 DI 110]

Geometria cartesiana no LOGO

A geometria da tartaruga baseia-se nas propriedades intrínsecas das figuras. É o oposto do que ocorre na geometria cartesiana, em que as figuras são definidas em relação a um sistema externo de coordenadas ortogonais. Os movimentos da tartaruga são determinados pelo seu próprio avanço — segundo um número fixo de unidades — a partir de sua posição inicial na tela. Para desenhar um triângulo, por exemplo, a rotina correta é REPITA 3[FR 50 DI 120], e não REPITA 3[FR 50 DI 60], que resulta na metade de um hexágono. Isso porque o ângulo a ser considerado é o formado com a trajetória da tartaruga, isto é, 120°, em vez de 60°. Na geometria cartesiana, a tela é imaginada como uma rede quadriculada, com unidades dispostas em um eixo vertical e outro horizontal. Cada ponto da rede possui um valor numérico específico e os movimentos são definidos em relação a esses referenciais absolutos. Apesar disso, também é possível mover a tartaruga utilizando as coordenadas ortogonais.

Por exemplo, o comando DEFXY 20 30 move a tartaruga de sua posição inicial para o ponto (20,30). Acionando o comando COLORIDO (CL), a tartaruga traça uma linha à medida que se move. O comando SEMCOR (SC), ao contrário, faz com que a tartaruga se mova sem deixar rastro. A origem (0,0) fica no centro da tela.

As rotinas seguintes têm como resultado a mesma figura e ilustram a diferença entre a geometria da tartaruga e a geometria cartesiana:

```
AP QUADRADO1
REPITA 4 [FR 50 DI 90]
FIM
AP QUADRADO2
DEFXY 0 50
DEFXY 50 50
DEFXY 50 0
DEFXY 0 0
FIM
```

Ao digitar QUADRADO1 ou QUADRADO2, depois de retornar ao modo desenho (comando DESENHE), a mesma figura será traçada na tela. Mas o que aconteceria se você quisesse girar os dois quadrados em 30°? No primeiro caso, a instrução DI 30 QUADRADO1 será suficiente para se obter a rotação da figura. A segunda rotina, contudo, precisa ser reescrita (pois opera com coordenadas absolutas na tela), o que não é tarefa simples. Mas há ocasiões em que o emprego da geometria cartesiana no LOGO facilita muito certas tarefas. Nesses exemplos, você perceberá que, para traçar linhas, DEFXY é muito mais rápido do que FRENTE.

Outra característica da geometria da tartaruga é a sua "miopia". A tartaruga realiza apenas um movimento por vez. Ao traçar as figuras, ela cumpre uma série de pequenos "passos". Colocando-se no lugar da tartaruga é mais fácil perceber as condições necessárias para desenhar uma figura circular. Seria preciso mover-se um pouco para a frente e depois virar-se também um pouco, repetindo a seqüência muitas vezes. Na linguagem LOGO, isso se traduz por:

```
AP CIRCULO
REPITA 360 [FR 1 DI 1]
FIM
```

Embora produza o resultado esperado, trata-se de uma rotina muito lenta. Aumenta-se sua velocidade tornando a tartaruga invisível, mediante o comando SEMT (ST). O comando MOSTRET (MT) faz com que ela retorne à tela. Na verdade, o que se pretende com essa rotina é desenhar um polígono de 360 lados. Num polígono assim, os lados são tão pequenos que parecem compor uma linha curva contínua. Tal quantidade de lados, no entanto, é muito maior do que a necessária para dar a ilusão de um círculo — um polígono de 36 lados é mais do que suficiente. Assim, em muito menos tempo, traça-se um círculo aceitável com a rotina:

```
AP CIRCULO
FIM
```

Tente agora desenvolver rotinas que produzam um semicírculo, um quarto de círculo e um sexto de círculo. Em seguida, combine duas dessas figuras para formar uma pétala.

Voltando ao lugar da tartaruga, como você se moveria para desenhar uma espiral terminando num ponto, em vez de apenas circundá-lo? Como no círculo, seria preciso avançar um pouco antes de virar; nesse caso, porém, a cada volta você se viraria um pouco mais do que na anterior (ou, o que é a mesma coisa, percorreria distância menor a cada volta). A rotina a seguir, embora um pouco longa, ilustra esse princípio:

```
AS ESPIRAL
FR 10 DI 10 FR 10 DI 20 FR 10 DI 30
FR 10 DI 40 FR 10 DI 50 FR 10 DI 60
FR 10 DI 70 FR 10 DI 80 FR 10 DI 90
FIM
```

Junção das partes

Por ser uma linguagem extensível, o LOGO permite definir rotinas a partir de outras. O quebra-cabeça chinês Tangram, um quadrado recortado em sete peças geométricas, serve de ilustração. Essas peças se combinam de várias maneiras, compondo figuras.

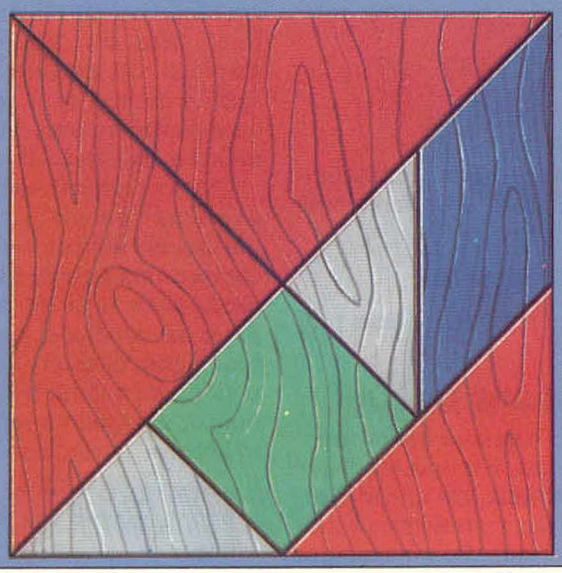
Para fazer a figura de um cachorro, por exemplo, começa-se pela criação de rotinas LOGO para cada parte básica. Tais "rotinas de partes" serão depois incorporadas a outra rotina mais ampla, chamada CAO. Como a tartaruga precisa ser corretamente posicionada para desenhar

Figuras simples

Ao lado encontram-se as rotinas LOGO para desenhar as sete peças básicas do quebra-cabeça Tangram. Há também um exemplo de rotina para a montagem da figura de um cachorro. Sob o nome CAO, essa rotina começa pelo traçado do triângulo que representa a perna traseira do animal.

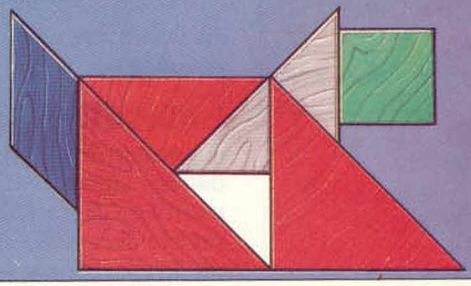
Rotinas Tangram

```
AP QUADRADO
  REPITA 4 [FR 25 DI 90]
FIM
AP PAR
  REPITA 2 [FR 25 DI 45 FR 35 DI 135]
FIM
AP TRI1
  FR 25 DI 135 FR 35 DI 135 FR 25 DI 90
FIM
(Há dois triângulos desse tamanho.)
AP TRI2
  FR 35 DI 135 FR 50 DI 135 FR 35 DI 90
FIM
AP TRI3
  FR 50 DI 135 FR 71 DI 135 FR 50 DI 90
FIM
(Há dois triângulos desse tamanho.)
```



Programa do cão

```
AP CAO
  TRI3 MOVE1 PAR MOVE2 TRI2 MOVE3
  TRI1 MOVE4 TRI3 MOVE5 TRI1 MOVE6
  QUADRADO MOVE7
FIM
AP MOVE1
  SC FR 15 ES 45 CL
FIM
AP MOVE2
  SC DI 45 FR 35 ES 45 VO 35 CL
FIM
AP MOVE3
  SC ES 45 VO 25 CL
FIM
  AP MOVE4
  SC DI 90 VO 25 CL
FIM
AP MOVE5
  SC FR 50 DI 45 CL
FIM
AP MOVE6
  SC FR 25 DI 135 FR 5 ES 90 CL
FIM
AP MOVE 6
  SC FR 25 DI 135 FR 5 ES 90 CL
FIM
AP MOVE7
  SC ES 90 FR 5 DI 45 VO 25 DI 45
  VO 50 ES 90 VO 50 CL
FIM
```



cada uma das peças, é necessário desenvolver também outras rotinas especiais — de MOVE1 até MOVE7.

Seria fácil criar esse desenho apenas interligando os comandos numa longa rotina. Entretanto, o método aqui aplicado é o da ordem "invertida": trata-se de dividir o problema em várias partes e solucioná-las uma por vez. Para o programador, o método é vantajoso, pois lhe permite criar uma rotina contendo sub-rotinas ainda não definidas. A rotina principal só pode ser rodada quando as sub-rotinas forem especificadas ou substituídas por pseudo-rotinas.

A rotina CAO, no exemplo, foi a primeira a ser escrita, seguida pelas sub-rotinas escritas separadamente e, por fim, pelas rotinas de posicionamento da tartaruga. Após desenvolver uma nova sub-rotina, rodava-se a rotina CAO para assegurar o encaixe perfeito de todos os seus componentes. Quando chegava a uma rotina MOVE ainda não escrita, o LOGO parava e enviava uma mensagem de erro. A partir do desenho, determinava-se com facilidade quando o erro estava na rotina principal ou na última rotina MOVE.

O exemplo demonstra que nem as sub-rotinas nem a rotina principal alteram o estado da tartaruga. Isso significa que ela retorna à posição em que estava antes da execução da rotina. Essas rotinas, chamadas "transparentes", facilitam a tarefa de encadear rotinas para elaboração de desenhos mais complexos. Considere a rotina CAO: sabendo que, depois de traçar uma das peças, a tartaruga retornará à posição inicial, é desnecessário conhecer o funcionamento interno das sub-rotinas para montar a rotina principal. E, com a rotina CAO transparente, ficou mais fácil incorporá-la em outra rotina — por exemplo, uma que multiplicasse a figura do cachorro na tela.

Área de trabalho LOGO

A memória operacional do LOGO consiste em uma lista de modos, cada um de cinco bytes. Depois de carregar o LOGO, ficam disponíveis entre 1.000 e 3.000 módulos, dependendo do modelo do micro. Os módulos são ocupados à medida que se gravam as rotinas. Outros módulos ainda podem ser usados enquanto as rotinas são executadas, ou no caso de se empregarem variáveis (que serão analisadas em outro artigo).

As rotinas gravadas constituem a área de trabalho. Faz-se a verificação das rotinas mantidas na área de trabalho mediante o comando LISTAR ROTINA (abreviado LI ROTINA). Para examinar uma rotina específica, use LISTAR (LI) — por exemplo, LI QUADRADO. Se a rotina não é mais necessária, esvazia-se o espaço de trabalho acionando o comando SEM. Por exemplo, SEM QUADRADO eliminará da memória a rotina QUADRADO. A eliminação de uma rotina libera o módulo por ela ocupado. O LOGO assinalará esses módulos livres, mas não os colocará de imediato na lista dos utilizáveis. Em lugar disso, continuará operando com os módulos anteriormente disponíveis, até que todos tenham sido usados. Só depois o LOGO pesquisará na memória os módulos liberados, formando com eles uma nova lista de módulos disponíveis. Esse processo é conhecido como "limpeza" e nele o LOGO parece hesitar por um ou dois segundos de vez em quando.

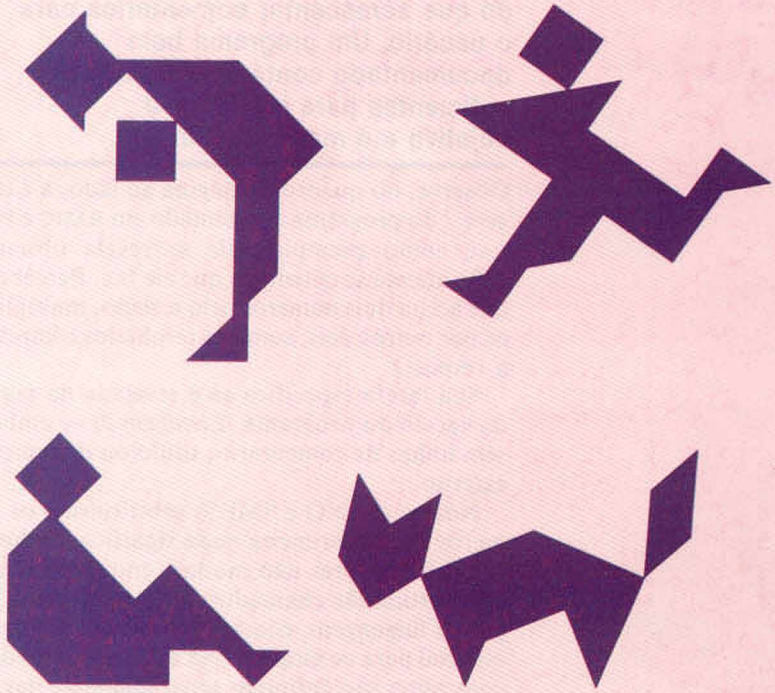
Exercício 2

A) Desenvolva rotinas que tracem as figuras do Tangram abaixo. É preciso antes descobrir como se dispõem as diferentes peças em cada figura.

B) Crie a rotina que desenha uma casa — basta um triângulo eqüilátero sobre um quadrado.

C) Escreva a rotina para um quadriculado com 5 x 5 quadrados.

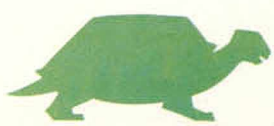
D) Descreva a rotina usada para a estrela de seis pontas, de modo que seja formada de sub-rotinas.



Rotinas de armazenamento

Para registrar rotinas em disco de modo permanente, grava-se o espaço de trabalho sob a forma de arquivo. Supondo que DESENHOS seja o nome de um arquivo, digita-se GRAVE "DESENHOS (note que as aspas vêm apenas antes e não depois do nome-arquivo). O espaço de trabalho não será afetado por essa operação. O arquivo pode ser carregado por meio do comando LEIA "DESENHOS, que faz com que os procedimentos no arquivo sejam definidos e acrescentados à área de trabalho em uso. Quando se atribui a um procedimento o mesmo nome de outro já gravado, o novo procedimento substitui o anterior.

Outros comandos úteis para a manipulação de discos são VERDISCO e SEMARQUIVO. O primeiro fornece uma lista de todos os arquivos no disco. SEMARQUIVO "DESENHOS, por exemplo, elimina o arquivo DESENHOS do disco.



CUIDADOS COM O ESTILO

Documentar o programa é mais do que acrescentar comentários para o usuário. Um programa bem documentado contém informações suficientes para indicar seu objetivo e o modo de atingi-lo.

Observe, no quadro da página ao lado, a *Listagem 1* do programa apresentado em BASIC e PASCAL como exemplo. Ele se revela obscuro; dificilmente se entende o que ele faz. Percebe-se que aceita dois números pelo teclado, multiplica-os por outros dois, soma os resultados e imprime a resposta.

Sua tarefa específica só é revelada na segunda versão do programa *(Listagem 2)* — embora sem linhas de comentário, título ou documento externo.

Nomes (SANO e SMES) substituíram os números que na primeira nada significam. Números cujos valores não mudam enquanto o programa roda são chamados de "constantes". Algumas linguagens, como a PASCAL, têm notação especial para constantes. Na *Listagem 2,* as duas constantes são definidas separadamente das variáveis. Outras linguagens, como o BASIC, não fazem essa separação: as linhas 10 e 20 do programa em BASIC usam variáveis para definir as constantes. Dar nomes às constantes só interessa quando seu uso é freqüente; caso contrário, basta inserir no programa alguns comentários — linhas iniciadas por REM (de REMARK, ou seja, "comentário").

Uma segunda diferença ressalta: todos os nomes confusos das variáveis, representados apenas por uma letra, foram substituídos por palavras inteligíveis. ANOS (em vez de A), IDADESEG (em vez de E) e outras foram selecionadas por terem menos de dez caracteres e os dois primeiros serem diferentes entre si.

Em geral, constitui boa prática dar às variáveis nomes relacionados com o papel que desempenham no programa. Por exemplo: pode-se chamar de CONTADOR um contador de loops e atribuir seu primeiro e seu último valores a constantes ou a variáveis de valores predeterminados. Assim, um loop como FOR J = 1 TO 10-NEXT J representa-se por FOR CONTADOR = A TO M-NEXT CONTADOR.

Nomes longos de variáveis exigem mais digitação e memória, mas resultam em programas inteligíveis e facilmente debugáveis (depuráveis de seus erros).

Se a linguagem pela qual você optou usa apenas os dois primeiros caracteres de uma variável para identificá-la — o que ocorre com quase todas as versões do BASIC —, escolha nomes em que os dois primeiros caracteres diferem entre si. Isso porque, no caso, nomes longos de variáveis como CODNUM e COMP serão iguais para o computador.

Outra grande diferença entre as listagens-exemplo: a segunda usa mensagens bem explícitas ao pedir as entradas e ainda acrescenta uma explicação para cada saída (a linha PRINT, em BASIC, e a WRITE, em PASCAL).

Com isso, o programa torna-se mais inteligível para o usuário e para o próprio programador.

A diagramação do programa

Recursos simples, como deixar linhas em branco, alternar letras maiúsculas e minúsculas e alinhar as instruções por grupos, transformam uma impenetrável massa de símbolos em exposição lógica, organizada e legível.

Formatar um programa para a tela ou para a impressora é fundamental quando ele se vale de construções em loop (FOR-NEXT, WHILE-WEND, REPEAT-UNTIL) e sobretudo quando os loops contêm outros loops, pois a posição das instruções indica onde começa ou termina cada seqüência de instruções executada repetitivamente até que ocorra certa condição.

O BASIC permite pouca escolha quanto à apresentação do programa. Linguagens compiladas, como a PASCAL, são bem mais flexíveis. Nelas, escrevem-se os programas com o auxílio de um editor de texto. Já em BASIC — salvo na versão para o TRS-80 —, os recursos para edição mostram-se elementares.

Os comentários são o principal meio de documentação interna do programa. O BASIC vale-se de linhas de comentários: a palavra REM as inicia e o interpretador ignora tudo o que encontrar até o próximo marcador de fim de linha (: ou RETURN). Em outras linguagens — PASCAL, PL/1, PROLOG etc. —, os comentários ficam entre os sinais /* e */ ou { e }. Com esse sistema, os comentários podem ocupar mais de uma linha. Mas, se você esquecer o */, o resto do programa será considerado um comentário e ignorado pelo compilador.

Acrescente comentários sempre que uma explicação for necessária: quando começar programas ou novas sub-rotinas, definir constantes ou funções, escrever instruções complexas etc.

Por outro lado, evite comentários longos ou prolixos. Muitas vezes, basta um lembrete curto e direto, colocado antes de um bloco de instruções complexas. Coloque-o dentro da sub-rotina apenas quando sua presença não interferir na

compreensão da estrutura lógica do programa. A *Listagem 3* constitui um exemplo do bom uso de comentários.

Documentação externa

Manuais e especificações escritas são de produção difícil e tediosa. Pesquisas européias demonstraram que, em geral, os programadores só consultam documentação externa como último recurso. No entanto, quando bem usada, ela poupa esforços consideráveis.

As linguagens de quinta geração pretendem aumentar a produtividade do programador também por meio da criação automática de documentação, o que se pode conseguir pelo uso de informações selecionadas da fase de projeto do programa.

E uma das melhores maneiras de documentar seus programas, por enquanto, é usar o mesmo recurso. Mantenha sempre uma pasta para cada programa, enquanto você escreve. Guarde ali todas as anotações que fizer durante o projeto do programa — inclusive fluxogramas e esboços de algoritmos. É fundamental arquivar a versão final do fluxograma usado para escrever as instruções.

Se você tem uma impressora, guarde também uma listagem do programa acabado. Em nossa versão completa do programa-exemplo, o primeiro comentário contém o nome e a data. Essa precaução permite, sempre que se modificar o programa, a atualização da data para se saber qual é a última versão.

Registre

Linha TRS-80

HEXADECIMAL PARA DECIMAL

Se você tem um CP 500 ou um CP 300 versão cassete, deve sentir falta de uma rotina para transformar números hexadecimais (de base 16) em decimais (de base 10). Endereços de memória e certos valores para POKE são normalmente fornecidos em notação hexadecimal.

Para transformá-los em decimais, você precisa ter uma calculadora à mão e realizar uma série de contas. Mas com este pequeno programa em BASIC entra-se com o número em hexadecimal e obtém-se imediatamente seu equivalente decimal.

```
10 INPUT "TECLE NUMERO HEXA: ";A$
20 P=4096:R=0:FOR X=1 TO 4:N=ASC(MID$
   (A$,X,1)):IF N<48 OR N>70 OR (N<65
   AND N>57) OR LEN (A$)<>4 THEN PRINT
   "ERRO":GOTO 10
30 B=N-48+7*(N>64):R=R+B*P:P=P/16:NEXT X
40 PRINT "DECIMAL: ";R:GOTO 10
```

Lembre-se de que a notação hexadecimal usa os dez dígitos numéricos e as letras de A até F. Entre com números de quatro dígitos de 0000 até FFFF para obter o resultado decimal de 0 até 65.535.

Documentação apropriada

Listagem 1

BASIC

```
(A) 10 INPUT A,B
    20 C=A*31536000
    30 D=B*2592000
    40 E=C+D
    50 PRINT E
```

PASCAL

```
(B) PROGRAM ABCDE (INPUT,OUTPUT);
    VAR A,B,C,D,E:INTEGER;
    BEGIN
    READ(A,B);
    C:=A*31536000;
    D:=B*2592000;
    E:=C+D;
    WRITELN(E);
    END.
```

Listagem 2

BASIC

```
(A) 10 ANO=31536000
    20 MES=2592000
    30 PRINT "ENTRE SUA IDADE (ANOS DEPOIS MESES SEPARADOS
       POR UMA VIRGULA" ;
    40 INPUT NANOS,NMESES
    50 SANO=NANOS*ANO
    60 SMES=NESES*MES
    70 IDADESEGS=SANO+SMES
    80 PRINT "SUA IDADE EM SEGUNDOS E APROXIMADAMENTE
       ";IDADESEGS
```

PASCAL

```
(B) PROGRAM IDADESEGS (INPUT,OUTPUT);
    CONST
      ANO=31536000;
      MES=2592000;
    VAR
      NANOS,NMESES,SANO,SMES,IDADESEGS:INTEGER;
    BEGIN
      WRITE('ENTRE SUA IDADE (ANOS DPOIS MESES SEPARADOS POR
      UMA VIRGULA) ');
      READ (NANOS,NMESES);
      SANO:=NANOS*ANO
      SMES:=NMESES*MES;
      IDADESEGS:=SANO+SMES
      WRITELN('SUA IDADE EM SEGUNDOS E IDADESEGS):
    END.
```

Listagem 3

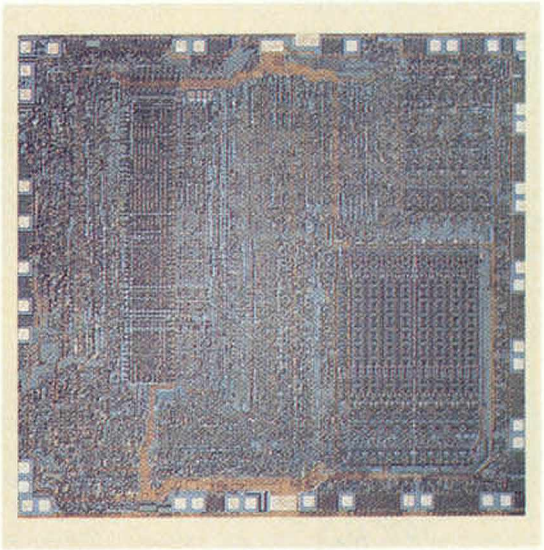
BASIC

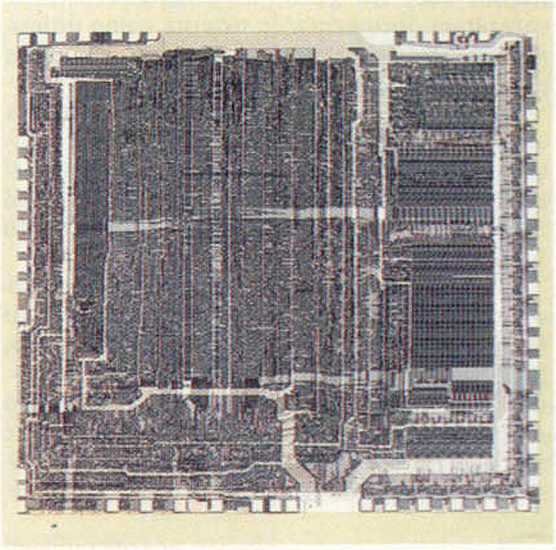
```
(A) 10  REM "IDADE EM SEGUNDOS " MAIO DE 1985
    20  REM INPUT'S A IDADE EM ANOS E MESES (A,M) E
    30  REM USA UMA CONVERSAO APROXIMADA (1 MES= 30 DIAS)
    40  REM PARA RETORNAR A IDADE EM SEGUNDOS.
    50  REM
    60  ANO=31536000:REM         SEGUNDOS EM 365 DIAS
    70  MES=2592000 :REM         SEGUNDOS EM 30  DIAS
    80  PRINT "ENTRE SUA IDADE (ANOS DEPOIS MESES SEPARADOS
        POR UMA VIRGULA ";
    90  INPUT NANOS,NMESES
    100 REM IDADE EM SEGUNDOS E' (IDADE EM ANOS * ANO EM
        SEGUNDO) MAIS (MESES DESDE ULTIMO ANIVERSARIO * MES
        EM SEGUNDO)
    110 SANO=NANOS*ANO
    120 SMES=NMESES*MES
    130 IDADESEGS=SANO+SMES
    140 PRINT "SUA IDADE EM SEGUNDOS E (APROXIMADAMENTE)
        ";IDADESEGS
```

PASCAL

```
(B) PROGRAM IDADESEGS (INPUT,OUTPUT);
    /*      MAIO DE 1985
        LE IDADE EM ANOS E MESES (ANO,MES) E USA UMA
        CONVERSAO APROXIMADA (MES= 30 DIAS) PARA
        RETORNAR IDADE EM SEGUNDOS.
    CONST
      ANO=31536000;       /*SEGUNDOS EM 365 DIAS*/
      MES=2592000;        /*SEGUNDOS EM 30  DIAS*/
    VAR NANOS,NMESES,SANO,SMES,IDADESEGS:INTEGER;
    BEGIN
      WRITE('ENTRE SUA IDADE (MESES E ANOS SEPARADOS POR UMA
            VIRGULA) ');
      READ(NANOS,NMESES);
      /* IDADE EM SEGUNDOS E (IDADE EM ANOS * ANO EM
      GUNDOS) MAIS (MESES DESDE O ULTIMO ANIVERSARIO *MES EM
      SEGUNDOS)       */
      SANO:=NANOS*ANO;
      SMES:=NMESES*MES;
      IDADESEGS:=SANO+SMES;
      WRITELN('SUA IDADE EM SEGUNDOS E ',IDADESEGS);
    END.
```

OS DISSIDENTES





Conjunto de chips
Microfotografias que ilustram a complexidade dos circuitos do microprocessador. À esquerda, uma foto altamente ampliada do bem-sucedido Z80 da Zilog; à direita, um chip recente da série Z8000. O Z8000, um chip de 16 bits, possui desenho mais denso e mais conexões em torno das bordas.

Em 1977, a Zilog lançou o Z80. Em poucos anos, esse microprocessador e o 6502 tornaram possível o que antes parecia ficção científica: um computador em cada domicílio.

A história da empresa americana Zilog Incorporated começa no início da década de 70, quando Frederico Faggin e Masatoshi Shima, funcionários da Intel (fabricante de microchips), saíram dessa companhia. Anteriormente, os dois haviam participado do desenvolvimento do microchip 8080A (considerado o primeiro "computador num chip"). Com essa experiência, começaram a trabalhar em novos desenvolvimentos do microprocessador.

O 8080 já era muito popular entre amadores e projetistas de computação; assim, Faggin e Shima decidiram desenhar o novo chip de modo que fosse compatível com o 8080, aproveitando deste a grande quantidade de software. Eles ampliaram o conjunto de instruções (a lista de comandos em código de máquina presente no microprocessador) pela introdução de registros extras, códigos de operação de 2 bytes e outras técnicas. Seu microprocessador — o Z80 — constitui um aperfeiçoamento considerável.

Essa revolução no hardware coincidiu com mudanças no software. Em 1972, Gary Kildall e John Torode escreveram o programa Control Program/Monitor (CP/M), que possibilitava a um microprocessador manipular o recém-lançado disco flexível.

Como Kildall era consultor da Intel, o programa foi projetado para rodar no 8080 e no 8055. O CP/M tornava-se rapidamente o sistema de manipulação de discos dominante para os microcomputadores. Com seu potente microprocessador compatível com o 8080, a Zilog estava estrategicamente situada e podia tomar a dianteira na corrida dos softwares para o CP/M.

Frank de Weeger
Presidente da Zilog

Programação difícil

O caminho da Zilog não foi tão suave nos anos subseqüentes. Embora o Z80 ainda vendesse bem em meados da década de 80 — quando a companhia produzia cerca de 1 milhão de unidades por mês —, as tentativas de adaptar o chip para o mercado de 16 bits não obtiveram a reação esperada.

A primeira tentativa da Zilog com um processador de 16 bits foi o Z8000. Embora reconhecido como dispositivo muito potente, com amplo conjunto de instruções e grande número de registros, o chip revelou-se extremamente difícil de se programar.

O Z8000 encontrou ainda vários outros obstáculos. Era compatível com o Z80000 (ou Z80K), de 32 bits, mas não com o Z80. Não podia, portanto, aproveitar a amplitude e a variedade de programas escritos para o Z80 nos anos precedentes. Os fabricantes interessados nos chips de 16 bits para seus equipamentos voltaram-se, pouco a pouco, para outros microchips, com menos exigências.

Com o Z8000 pouco popular no mercado de micros, a Zilog voltou à mesa de projetos a fim de estudar o processador Z800 de 16 bits, compatível com o Z80. Paralelamente, a empresa de computadores Commodore anunciou que sua nova série de máquinas para uso comercial seria suprida com o Z8000.

Acerte a mira
O navio pode ser atingido em três regiões, e os pontos variam de 0 a 30. O resultado será bom se você conseguir 150 pontos ou mais.

```
1128 GOTO 1110
1130 INVERSE: HOME:VTAB 3:HTAB 11:PRINT
     "*** ATAQUE AEREO ***:NORMAL:VTAB 7
1140 PRINT "VOCE TEM 15 BOMBAS."
1145 PRINT: PRINT "A VELOCIDADE DO NAVIO
     VARIA, PORTANTO TOME CUIDADO PARA
     NAO DESPERDICAR BOMBAS."
1150 PRINT: PRINT: PRINT TAB (11)" B O A
     S O R T E"
1155 NORMAL
1160 VTAB 23
1165 INVERSE
1170 PRINT "TECLE <RETURN> PARA
     CONTINUAR"
1175 IF PEEK (-16384)=141 THEN POKE
     -16368,0:GOTO 1180
1178 GOTO 1160
1180 NORMAL:RETURN
2000 REM *** CRIAR VARIAVEIS ***
2020 TS=0
2030 SL=15
2050 GR:COLOR=2
2060 HLIN 0,39 AT 39
2080 HLIN 0,39 AT 38
2090 AX=0:SX=33:SS=1
2100 RETURN
3000 REM *** O JOGO ***
3030 HOME
3035 INVERSE
3040 VTAB 22:CALL -958:PRINT "TIROS
     FALTANDO";:NORMAL:PRINT" ";SL;" ";:
     INVERSE: PRINT "PONTOS":NORMAL
     PRINT " ";TS
3050 GOSUB 3090:REM AVIAO
3060 GOSUB 3130:IF SL=0 THEN RETURN
3070 GOSUB 3270:REM NAVIO
3080 GOTO 3050
3090 COLOR=0:HLIN AX,AX+6 AT 2:HLIN
     AX+1,AX+6 AT 1:PLOT AX+6,0
```



```
3100 REM
3110 AX=AX-1: IF AX<0 THEN AX=33
3120 COLOR=4:HLIN AX,AX+6 AT 2:HLIN
     AX+1,AX+6 AT 1:PLOT AX+6,0:RETURN
3130 IF FF THEN 3160
3140 IF PEEK (-16384)=<128 THEN RETURN
3150 FF=1:POKE -16368,0:FX=AX+3:FY=2
3160 COLOR=0:PLOT FX,FY
3170 FY=FY+1
3180 IF SCRN (FX,FY)=0 THEN
     COLOR=13:PLOT FX,FY:RETURN
3190 IF SCRN (FX,FY)=2 THEN 3230
3200 TS=TS+(38-FY)*10
3210 SC=0:COLOR=0:HLIN SX,SX+6 AT
     37:HLIN SX+3, SX+5 AT 36 : PLOT
     SX+4,35
3220 SX=33:SS=1:SC=0
3230 COLOR= 2:HLIN FX-1,FX+1 AT 37: PLOT
     FX-2,36:PLOT FX,36: PLOT FX+2,36 :
     PLOTFX-3,35:PLOT FX,35:PLOT FX+3,35
3240 COLOR= 0: HLIN FX-1,FX+1 AT 37:
     PLOT FX -2,36: PLOT FX,36: PLOT
     FX+2,36:PLOT FX-3,35:PLOT FX,35:
     PLOT FX+3,35
3250 FF=0:SL=SL-1:POKE -16368,0
3255 INVERSE
3260 VTAB 22: CALL -958:PRINT "TIROS
     FALTANDO";:NORMAL: PRINT " ";SL;"
     ";:INVERSE: PRINT "PONTOS";:NORMAL
     PRINT  " ";TS
3270 SC=SC+1:IF SC=SS THEN RETURN
3280 SC=0:COLOR=0:HLIN SX,SX+6 AT 37:
     HLIN SX+3,SX+5 AT 36:PLOT SX+4,35
3290 SX=SX+1: IF SX=> 33 THEN
     SX=0:SS=INT( RND(1)*3)+1:SC=SS
3300 COLOR=1:HLIN SX,SX+6 AT 37:HLIN
     SX+3,SX+5 AT 36: PLOT
     SX+4,35:RETURN
4000 REM *** FIM DE JOGO ***
4010 FOR N= 1 TO 3000:NEXT N
4020 TEXT: HOME: VTAB 3:HTAB 7: INVERSE:
     PRINT "*** ATAQUE AEREO ***"
4025 NORMAL
4030 PRINT:PRINT:PRINT "O JOGO
     ACABOU.":PRINT:PRINT:PRINT "VOCE
     OBTEVE ";:INVERSE:PRINT TS;:NORMAL:
     PRINT "PONTOS QUE E' ";
4040 IF TS =<25 THEN PRINT "HORRIVEL ":
     RETURN
4050 IF TS =<50 THEN PRINT "PESSIMO!!":
     RETURN
4060 IF TS =<75 THEN PRINT "RUIM!!! ":
     RETURN
4070 IF TS =<100 THEN PRINT "MEDIO! ":
     RETURN
4080 IF TS =<150 THEN PRINT "BOM!!! ":
     RETURN
4090 IF TS =<250 THEN PRINT "OTIMO!!!":
     RETURN
4100 IF TS =<450 THEN INVERSE: PRINT "":
     PRINT "FANTASTICO !!!":NORMAL:
     RETURN
```

INVASOR DO ESPAÇO

PARA LINHA SINCLAIR

O invasor extraterreno quer atingir seu canhão. Procure explodi-lo antes que ele chegue a Terra. Acertando-o quando ainda estiver bem alto, você ganhará mais pontos e evitará uma aproximação perigosa. O invasor pode tomar a Terra quatro vezes. A cada invasão, você perderá um canhão. Mas, fazendo 3.000 pontos, terá direito a bônus e mais um canhão. Observe os comandos: "8" movimenta o canhão para a direita; "5" move-o para a esquerda; com "O" você atira; e "C" reinicia o jogo. Para aumentar a dificuldade, basta mudar a linha 500 B = 3000 para um valor mais alto em B.

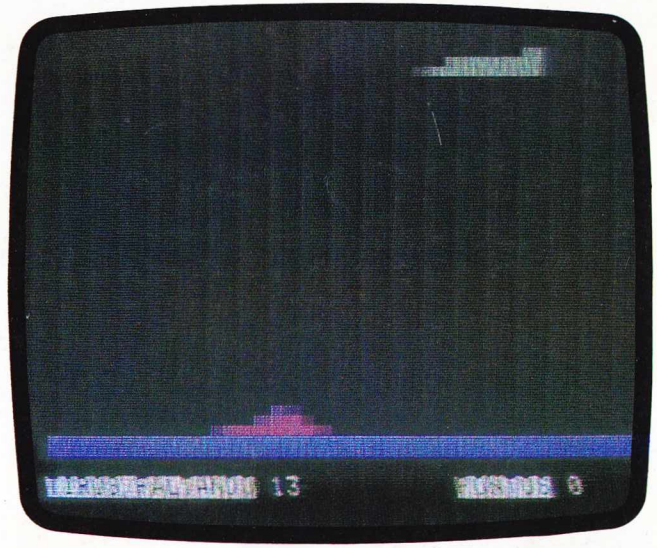
```
10   REM ********************
15   REM * INVASOR DO ESPACO *
20   REM ********************
30   GOTO 490
40   FOR L= U TO 19
50   LET X=Y + (INKEY$ = "8" AND X < 26)
     - (INKEY$= "5" AND X > 3)
55   REM *** CANHAO ***
60   PRINT AT 19,X;" I$** "
65   REM *** TIRO INVASOR ***
70   LET C = C + A (L)
80   PRINT AT L - 1,C -1;"    "
90   PRINT AT L,C;" O "
92   PRINT AT L,C;"    "
95   REM *** TIRO AO CANHAO ***
100  IF INKEY$="O" THEN GOSUB 160
110  NEXT L
120  LET F=F - 1
130  LET Z=0
140  IF F < 0 OR F = 0 THEN GOTO 895
150  GOTO 790
160  FOR M=17 TO L STEP -2
170  PRINT AT M,X + 1;"."
180  PRINT AT M + 2,X + 1;" "
190  NEXT M
200  PRINT AT M + 2,X + 1;" "
210  IF C = X + 1 THEN GOTO 230
220  RETURN
230  FOR K = 1 TO 20
240  FAST
250  SLOW
260  NEXT K
265  REM *** BUM ***
270  FOR K= 1 TO 5
280  PRINT AT L,C - 1;" BUM ";AT L,C -1;
     " BUM "; AT L,C - 1;"    "
290  NEXT K
295  REM *** BONUS ***
300  LET Z = 1
310  LET J = 10 * ( INT (1 / L (L + 10)
     * 1000))
```

Emoção garantida
As opções de comando e a possibilidade de dificultar a conquista de pontos garantem momentos de verdadeira emoção.

```
320  PRINT AT L,C - 1;J
330  LET H = C
340  LET P = P + J
350  GOSUB 880
360  IF P > B OR P = B THEN GOTO 390
370  LET U = U + 1
380  GOTO 790
390  LET J = 500 * F
400  PRINT AT 10,16;J
410  FOR K = 1 TO 30
420  PRINT AT 10,10;" BONUS "; AT 10,10;
     " BONUS ";AT 10,10;" BONUS "
430  NEXT K
440  IF U > 4 THEN LET U = U - 4
450  LET B = B + 3000
460  LET F = F + 1
465  REM A PROXIMA STRING TEM 11 ESPACOS
470  PRINT AT 10,10;"           "
480  GOTO 340
485  REM *** PRIMEIRA TELA ***
490  DIM A(25)
495  LET Z = 0
500  LET B = 3000
505  LET F = 4
510  LET P = 0
520  LET U = 1
530  LET L = 0
540  LET A = 0
550  LET X = 15
560  PRINT AT 3,4;"**INVASOR ESPACO **"
570  PRINT AT 10,1;"PRESS. QUALQUER
     TECLA PARA"
575  PRINT AT 12,10;"INICIAR"
600  FOR K = 1 TO 20
610  PRINT AT K,0;"  "; TAB( 31);"  "
620  NEXT K
625  REM AS QUATRO PROXIMAS STRINGS TEM
     31 ESPACOS
630  PRINT AT 1,1;"
                   "
631  PRINT AT 20,1;"
                   "
632  PRINT AT 1,1;"
                   "
635  PRINT AT 20,1;"
                   "
640  IF INKEY$ = "" THEN GOTO 630
644  REM *** AS ESTRELAS ***
645  PRINT "QUE A FORCA ESTEJA COM VOCE"
650  FOR K = 1 TO 20
660  SCROLL
670  PRINT TAB( RND * 32);"*"
680  SCROLL
690  PRINT TAB( RND * 32);"."
700  SCROLL
710  PRINT AT ( RND * 32);"+"
720  NEXT K
730  SCROLL
735  REM AS TRES PROXIMAS STRINGS TEM 32
     ESPACOS CADA UMA
740  PRINT "
        "
750  PRINT "
        "
760  PRINT "
        "
770  SCROLL
780  PRINT "
        "
790  GOSUB 880
800  IF Z = 1 THEN PRINT AT L,H -1;"  "
810  FOR I= 1 TO 20
820  LET A(I) = INT (RND * 3) - 1
830  LET X= X + (INKEY$ = "8" AND X<26)
     - ( INKEY$ = "5" AND X > 3)
840  PRINT AT 19,X;" I*** "
850  NEXT I
860  LET C = INT ( RND * 20) + 6
870  GOTO 40
875  REM *** QUADRO DE PONTOS ***
880  PRINT AT 21,2;P;" PONTOS";TAB21;F;
     " FOGUETES"
885  PRINT AT 20,0;"----------------------
     ----------"
890  RETURN
895  GOSUB 880
897  REM *** FINAL DO PROGRAMA ***
900  PRINT AT 10,10;" FIM DE JOGO "; AT
     10,10;" FIM DE JOGO "
901  REM *** NOVA PARTIDA ***
905  PRINT AT 12,4;"NOVA PARTIDA TECLE
     <C>"
910  IF INKEY$ = "C" THEN GOTO 1000
920  GOTO 900
1000 CLS
1010 RUN
```


ATAQUE AÉREO
PARA LINHA APPLE

Este jogo vai levá-lo ao oceano pilotando um avião de guerra. Sua missão é afundar todos os navios inimigos que atravessarem a rota do avião. Mas, cuidado! Os navios estarão cruzando o mar com velocidades variáveis e, além disso, atingi-los na proa ou na popa não vai afundá-los, apenas danificá-los. Por isso, calcule bem sua velocidade e a trajetória dos projéteis, a fim de acertá-los no centro, onde o número de pontos que você ganha é maior.
Depois de dez navios atingidos, você obterá sua contagem de pontos. Procure superar a marca anterior.



Navios velozes
A grande dificuldade deste jogo reside na velocidade variável dos navios, o que provoca, muitas vezes, um desperdício de bombas.

```
10   REM ATAQUE AEREO
80   GOSUB 1000:REM INSTRUCOES
90   GOSUB 2000:REM CRIA VARIAVEIS
100  GOSUB 3000:REM INICIO DE JOGO
110  GOSUB 4000:REM FIM DE JOGO
120  VTAB 21:HTAB 1:INVERSE
130  PRINT "VOCE QUER JOGAR NOVAMENTE ?
     (S/N) ";GET A$:NORMAL
140  IF A$="S" THEN GOTO 90
150  IF A$<> "N" THEN  GOTO 120
160  VTAB 21:HTAB 1: PRINT SPC(39):END
1000 REM *** INSTRUCOES ***
1030 TEXT:HOME
1035 INVERSE
1040 VTAB 3:HTAB 11:PIRNT "*** ATAQUE
     AEREO ***"
1045 NORMAL
1050 PRINT: PRINT: PRINT "VOCE E UM
     PILOTO DE COMBATE."
1060 PRINT: PRINT "PONTOS SAO OBTIDOS
     QUANDO UMA DAS SUAS BOMBAS, ACERTA
     O NAVIO, QUE TEM QUE AFUNDAR."
1070 PRINT
1080 PRINT "PARA SOLTAR UMA BOMBA APERTE
     QUALQUER TECLA. O NUMERO DE PONTOS
     OBTIDOS DEPENDERA DA PARTE DO NAVIO
     QUE FOI ATINGIDO."
1090 PRINT
1100 PRINT "A PARTE INFERIOR DO NAVIO =>
     10 PONTOS"
1105 PRINT "A PARTE SUPERIOR DO NAVIO =>
     20 PONTOS"
1108 PRINT "A CHAMINE DO NAVIO => 30
     PONTOS"
1110 VTAB 23
1120 PRINT "TECLE <RETURN> PARA
     CONTINUAR"
1125 IF PEEK (-16384)=141 THEN POKE
     -16368,0: GOTO 130
```